Beatriz da Costa Pinto Norberto

Licenciada em Ciência e Engenharia Informática

Linguagens para a Computação de Alto Desempenho, utilizadas no processamento de *Big Data*: Um Estudo de Mapeamento Sistemático

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Prof. Dr. Vasco Miguel Moreira do Amaral,
Prof. Auxiliar, Universidade Nova de Lisboa

Co-orientador: Prof. Dr. Miguel Carlos Pacheco Afonso Goulão,
Prof. Auxiliar, Universidade Nova de Lisboa

Júri

Presidente: Prof. Dr. Artur Miguel Andrade Vieira Dias
Arguente: Prof. Dr. Jácome Miguel Costa da Cunha
Vogal: Prof. Dr. Vasco Miguel Moreira do Amaral

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Janeiro, 2019

Linguagens para a Computação de Alto Desempenho, utilizadas no processamento de *Big Data*: Um Estudo de Mapeamento Sistemático

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Aos meus pais e irmã.
Muito obrigada por tudo.

Agradecimentos

Primeiramente, quero agradecer a todos os docentes que contribuíram para o meu percurso académico, fazendo-me crescer tanto a nível pessoal como profissional, com especial atenção aos meus orientador Prof. Dr. Vasco Amaral e co-orientador Prof. Dr. Miguel Goulão pela sua dedicação, disponibilidade e apoio, essenciais para a concretização desta dissertação.

Agradeço, também, a todos os membros do grupo de trabalho da *cHiPSet ICT COST Action* onde fui inserida, pela sua colaboração crucial para o desenvolvimento do trabalho.

À Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, pela formação de excelência oferecida aos alunos e, em particular, ao Departamento de Informática, que me acolheu ao longo destes últimos cinco anos.

A todos os meus amigos e companheiros de curso que, com a ajuda que me deram, facilitaram o enfrentar dos obstáculos que se atravessaram no meu caminho, ao longo desta jornada, e que, graças aos momentos que me proporcionaram, tornaram estes anos inesquecíveis.

Finalmente, um enorme obrigada à minha família, em especial, aos meus pais e irmã, não só pelo incentivo dado no decorrer desta caminhada, como pelo apoio incondicional, imprescindível para que consiga alcançar o sucesso na minha vida.

Resumo

Big Data são conjuntos de informação de alto Volume, Velocidade e/ou Variedade que exigem formas inovadoras e económicas de processamento, que permitem uma melhor percepção, tomada de decisões e automação de processos.

Desde 2002, a taxa de melhoria do desempenho em processadores simples diminuiu bruscamente. A fim de aumentar o poder dos processadores, foram utilizados múltiplos *cores*, em paralelo, num único *chip*. Para conseguir beneficiar deste tipo de arquiteturas, é necessário reescrever os programas sequenciais. O objetivo da Computação de Alto Desempenho (CAD) é estudar as metodologias e técnicas que permitem a exploração destas arquiteturas. O desafio é a necessidade de combinar o desenvolvimento de *Software* para a CAD com a gestão e análise de *Big Data*. Quando a computação paralela e distribuída é obrigatória, o código torna-se mais difícil. Para tal, é necessário saber quais são as linguagens a utilizar para facilitar essa tarefa.

Pelo facto da literatura existente sobre o tópico da CAD se encontrar muito dispersa, foi conduzido um Estudo de Mapeamento Sistemático (EMS), que agrega caraterísticas sobre as diferentes linguagens encontradas (categoria; natureza; perfis de utilizador típicos; eficácia; tipos de artigos publicados na área), no processamento de *Big Data*, auxiliando estudantes, investigadores, ou outros profissionais que necessitem de uma introdução ou uma visão panorâmica sobre este tema.

A pesquisa de artigos foi efetuada numa busca automatizada, baseada em palavras-chave, nas bases de dados de 8 bibliotecas digitais selecionadas. Este processo resultou numa amostra inicial de 420 artigos, que foi reduzida a 152 artigos, publicados entre Janeiro de 2006 e Março de 2018. A análise manual desses artigos permitiu-nos identificar 26 linguagens em 33 publicações incluídas. Sumarizei e comparei as informações com as opiniões de profissionais. Os resultados indicaram que a maioria destas linguagens são Linguagem de Propósito Geral (LPG) em vez de Linguagem de Domínio Específico (LDE), o que nos leva a concluir que existe uma oportunidade de investigação aplicada de linguagens que tornem a codificação mais fácil para os especialistas do domínio.

Palavras-chave: Computação de Alto Desempenho; *Big Data*; Linguagem de Domínio Específico; Linguagem de Propósito Geral; Estudo de Mapeamento Sistemático; Engenharia de *Software* Baseada em Evidências; Engenharia de *Software* Experimental

Abstract

Big Data are high-Volume, Velocity and/or Variety information sets that require innovative and cost-effective forms of processing that enable better insight, decision making and process automation.

Since 2002, the performance improvement rate on single processors has declined suddenly. In order to increase the power of the processors, multiple cores were used in parallel on a single chip. In order to benefit from this type of architectures, it is necessary to rewrite sequential programs. The goal of High Performance Computing (HPC) is to study the methodologies and techniques that allow the exploration of these architectures. The challenge is the need to combine software development for HPC with the management and analysis of Big Data. When the parallel and distributed computation is required, the code becomes harder. For this reason, it is necessary to know what languages should we use, in order to facilitate that task.

Because the existing literature on the topic of HPC is very dispersed, a Systematic Mapping Study (SMS) was conducted, which aggregates characteristics about the different languages found (category; nature; typical user profiles; effectiveness; types of articles published in the area), in Big Data processing, assisting students, researchers, or other professionals who need an introduction or a panoramic view on this subject.

The search for articles was carried out in an automated keyword-based search, in the database of 8 selected digital libraries. This process resulted in an initial sample of 420 articles, which was reduced to 152 articles, published between January 2006 and March 2018. The manual analysis of these articles allowed us to identify 26 languages in 33 included publications. I summarized the information extracted and compared it with the opinions of professionals in the area. The results indicated that most of these languages are General Purpose Languages over specific ones, which leads us to the conclusion that there is an opportunity for applied research on languages that make coding easier for domain specialists.

Keywords: High Performance Computing; Big Data; Domain Specific Language; General Purpose Language; Systematic Mapping Study; Evidence-Based Software Engineering; Empirical Software Engineering

Índice

Lista de Figuras

Lista de Tabelas

Glossário

Big Data	Conjuntos de informação de elevado Volume, Velocidade e/ou Variedade que exigem formas inovadoras e económicas de processamento, que permitem uma melhor percepção, tomada de decisões e automação de processos.
Engenharia de Software Baseada em Evidências	Área de pesquisa da Engenharia de *Software* que tem como finalidade melhorar a tomada de decisões relacionadas com o desenvolvimento e manutenção de *Software*, baseando-se nas melhores evidências encontradas.
Engenharia de Software	Área da Engenharia que se preocupa com todos os aspetos da produção de *Software*, desde as fases iniciais de especificação até à sua manutenção depois de ter sido utilizado.
Engenharia de Software Experimental	Área de pesquisa da Engenharia de *Software* que utiliza métodos empíricos como a análise de informação baseada em experiências e observações sobre um determinado tema.
Estudo Terciário	Uma revisão de estudos secundários relacionados com uma mesma questão de investigação.
Estudo de Mapeamento Sistemático	Estudo projetado para fornecer uma visão ampla, construída sobre questões gerais, de uma área de pesquisa para estabelecer se existem evidências de pesquisa sobre um determinado tópico e fornecer uma indicação dessas evidências.

Estudo Secundário	Um estudo que revê todos os estudos primários relacionados com uma questão de pesquisa específica, com o objetivo de sintetizar essas evidências.
Estudo Primário	Um estudo empírico que investiga uma questão de investigação específica.
Evidência	Síntese de estudos científicos sobre um tópico específico ou uma questão de pesquisa.
Linguagem de Propósito Geral	Linguagem de programação projetada para ser utilizada na escrita de *Software* numa ampla variedade de domínios de aplicação.
Linguagem de Domínio Específico	Linguagem adaptada a um domínio de aplicação específico que oferece anotações e abstrações apropriadas.
Meta-análise	Uma forma de estudo secundário, cuja síntese da pesquisa é baseada em métodos estatísticos quantitativos.
Protocolo de Revisão	Um plano que descreve a condução de uma qualquer revisão proposta.
Revisão Sistemática de Literatura	Uma forma de identificação, avaliação e interpretação de todas as pesquisas disponíveis relevantes para uma determinada questão de pesquisa, área ou fenómeno de interesse.

Acrónimos

CAD	Computação de Alto Desempenho.
EAP	Estrutura Analítica de Projetos.
EMS	Estudo de Mapeamento Sistemático.
ES	Engenharia de Software.
ESBE	Engenharia de Software Baseada em Evidências.
ESE	Engenharia de Software Experimental.
LDE	Linguagem de Domínio Específico.
LPG	Linguagem de Propósito Geral.

Capítulo 1

Introdução

1.1 Descrição e Contexto

Big Data são conjuntos de informação de alto Volume, Velocidade e/ou Variedade que exigem formas inovadoras e económicas de processamento, que permitem uma melhor percepção, tomada de decisões e automação de processos [Bak12; BL12; DM+15; SS13].

Desde 2002, a taxa de melhoria do desempenho em processadores simples diminuiu bruscamente. Com a finalidade de tornar mais eficiente, fiável e rápida a execução de programas que necessitam de recursos computacionais elevados, foram utilizados múltiplos *cores*, em paralelo, num único *chip*. Para conseguir beneficiar deste tipo de arquiteturas, é necessário reescrever (pelo menos, parcialmente) os programas sequenciais. O objetivo da Computação de Alto Desempenho (CAD, do inglês *High Performance Computing*) é estudar as metodologias e técnicas que permitem a exploração destas arquiteturas [LS09; Pac11].

Esta dissertação está integrada no âmbito do projeto *"Chipset - ICT COST Action IC1406 - High-Performance Modelling and Simulation for Big Data Applications (cHiPSet)"*, coordenado pela *Cracow University of Technology*. Este é um dos projetos em curso no *NOVA Laboratory for Computer Science and Informatics* (*NOVA LINCS*)[1] [Abob].

No contexto de *Big Data*, o desenvolvimento de *Software* para a CAD tem de ser combinado com a gestão e análise de *Big Data* [Bro+11; SS13]. A *COST* [Aboa] é uma rede de excelência internacional que tem a intenção de satisfazer a necessidade de coordenação entre os 37 países representados, para facilitar as interações entre os profissionais das duas áreas.

[1] O *NOVA LINCS*, hospedado no Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa, é uma unidade de investigação, líder em Portugal, na área da Ciência da Computação e da Engenharia.

1.2 Motivação

Big Data tornou-se uma das expressões mais utilizadas dos nossos tempos. Investigadores e profissionais dedicaram-se a desenhar ferramentas e técnicas inovadoras para acompanhar a rápida evolução e a crescente complexidade de grandes problemas científicos. *Big Data* foi definido como o modelo 3Vs [Bak12; BL12; DM+15; SS13]: *Big Data* são conjuntos de informação de elevado volume, velocidade e/ou variedade que exigem formas inovadoras e económicas de processamento, que permitem uma melhor percepção, tomada de decisões e automação de processos.

Com a finalidade de aumentar o poder dos processadores, foram utilizados múltiplos *cores*, em paralelo, num único *chip*. Para conseguir beneficiar deste tipo de arquiteturas, é necessário reescrever os programas em série, ou seja, aqueles programas que foram escritos para correr em processadores simples. O objetivo da CAD é estudar as metodologias e técnicas que permitem a exploração destas arquiteturas [LS09; Pac11].

O desafio da computação científica, no contexto de *Big Data*, é a necessidade de combinar o desenvolvimento de *Software* para a CAD com a gestão e análise de *Big Data* [Bro+11; SS13].

A necessidade de especialistas de desenvolvimento de *Software* específico para a CAD implica a formação de profissionais que conheçam as linguagens existentes que melhor suportem este tipo de computação. Infelizmente, a literatura científica existente encontra-se dispersa, pelo que seria útil existir um documento que agregasse esse tipo de informação. Estudantes, investigadores, ou outros profissionais que necessitem de uma introdução e uma visão geral sobre as linguagens disponíveis para a CAD beneficiariam de um documento que funcionasse como ponto de entrada fornecendo uma visão geral e apontadores úteis para ajudar a decidir quais são as linguagens mais promissoras a explorar, num determinado contexto.

Com base em pesquisas efetuadas tanto por mim como por outros elementos da *cHiPSet ICT COST Action*, em diversas plataformas eletrónicas como *ACM Digital Library*, *Elsevier Science Direct* e *Springer Link*, concluiu-se que, até ao momento, não existem estudos literários que agreguem, de modo sistemático, caraterísticas sobre as diferentes linguagens encontradas, no processamento de *Big Data*. Tal documento seria uma contribuição para o corpo de conhecimento que existe na área da CAD, consolidando a informação do estado da arte.

A fim de resolver esta problemática, foi realizado um Estudo de Mapeamento Sistemático (EMS, do inglês *Systematic Mapping Study*), isto é, um estudo projetado para fornecer uma visão ampla, construída sobre questões gerais, de uma área de pesquisa, para estabelecer se existem evidências de pesquisa sobre um determinado tópico e fornecer uma indicação dessas evidências [KC07; Pet+08a; Pet+08b]. Este EMS é uma tarefa fundamental de um dos grupos de trabalho da *cHiPSet ICT COST Action*. A minha função é a de coordenadora do estudo, tendo um papel executivo em todas as tarefas constituintes do mesmo. Em algumas tarefas, houve o envolvimento de colegas da *cHiPSet ICT COST Action*, para fazer face à escala do estudo a realizar. O EMS visa responder às seguintes questões principais, já validadas por profissionais da área:

1. "Quais são as categorias de linguagens em utilização para a CAD?" (Se a linguagem é uma LDE (do inglês *Domain Specific Language*), isto é, uma linguagem adaptada a um domínio de aplicação específico que oferece anotações e abstrações apropriadas; uma LPG (do inglês *General Purpose Language*), sendo uma linguagem de programação projetada para ser utilizada na escrita de *Software* numa ampla variedade de domínios de aplicação; uma LDE embebida numa outra LDE; uma LDE embebida numa LPG);

2. "Qual é a natureza das linguagens para a CAD?";

3. "Quais são os perfis de utilizador típicos para as linguagens?";

4. "Quão eficazes são as linguagens?";

5. "Que tipos de artigos são publicados na área de modelos de programação para a CAD?"

Para a criação do EMS, foram cumpridas todas as etapas do processo de revisão, sendo estas: o planeamento do EMS, onde foi identificada a necessidade da sua criação, especificadas as questões a pesquisar e desenvolvido um protocolo de revisão; a sua condução, passando, resumidamente, pela identificação dos estudos relacionados com as questões propostas, a extração de informação importante dos mesmos e a sumarização desses dados; a respetiva elaboração do EMS, nomeadamente, a sua escrita e divulgação [KC07]. Para além destas fases, foi realizado um questionário, disponível no Apêndice A, a pessoas especializadas na área da CAD, procurando saber:

1. As linguagens que utilizam atualmente;

2. As razões que os fazem utilizar essas linguagens em relação às restantes que conhecem;

3. As suas vantagens;

4. O quão eficazes são consideradas essas linguagens;

5. As ferramentas de suporte que são conhecidas;

6. As principais dificuldades sentidas na sua utilização.

Este questionário teve como objetivo confrontar a opinião de especialistas, tendo em conta a sua experiência na área, com a informação disponibilizada nos estudos.

1.3 Objetivo do Trabalho

O objetivo deste trabalho foi criar um EMS sobre as linguagens utilizadas na CAD, no processamento de *Big Data*, tendo sido analisados todos os estudos primários encontrados e resumidos os seus conteúdos, para conseguir responder às questões de pesquisa formuladas (quais são as categorias de linguagens em utilização para a CAD, qual é a natureza das linguagens para a CAD, quais são os perfis de utilizador típicos para as linguagens especificadas, quão eficazes são essas linguagens, que tipos de artigos são publicados na área de modelos de programação para a CAD).

Esta análise sistemática identifica as diferentes linguagens e os estudos relevantes a elas associados, e retira conclusões gerais sobre as suas caraterísticas, nas respostas dadas.

Durante o desenvolvimento do EMS, não só participei, ativamente, em todas as etapas do processo de revisão, como coordenei todo o estudo, tendo um papel executivo em todas as tarefas constituintes do mesmo.

1.4 Principal Contribuição Prevista

A principal contribuição que provém do desenvolvimento desta dissertação é apoiar estudantes, investigadores, ou outros profissionais que necessitem de uma introdução ou uma visão geral sobre as linguagens disponíveis para a CAD, no processamento de *Big Data*, no momento de decisão sobre quais são as mais promissoras a explorar, num determinado contexto.

Para tal, foi criado um EMS, com apontadores para estudos primários mais especializados, onde foram apresentadas as caraterísticas das linguagens identificadas através da nossa procura (categoria; natureza; perfis de utilizador típicos; eficácia; tipos de artigos publicados na área).

1.5 Estrutura do Documento

Este documento está estruturado da seguinte maneira:

- No **Capítulo 1**, é descrito e contextualizado o trabalho (Secção 1.1), seguido da motivação para a sua elaboração (Secção 1.2), objetivo da dissertação (Secção 1.3) e a principal contribuição prevista (Secção 1.4);

- No **Capítulo 2**, o leitor é enquadrado no tema da dissertação. Na Secção 2.1, é introduzido o conceito da Computação de Alto Desempenho (CAD), no processamento de *Big Data*. Na Secção 2.2, são definidas e comparadas as Linguagens de Domínio Específico (LDE) e as Linguagens de Propósito Geral (LPG). Seguidamente, é introduzida a Engenharia de *Software* e as áreas de Engenharia de *Software* Experimental (Subsecção 2.3.1) e Baseada em Evidências (Subsecção 2.3.2). Finalizando o capítulo, na Secção 2.4, é definido um EMS e o outro tipo comum de estudo secundário existente (Revisão Sistemática de Literatura) e, por fim, comparados os dois tipos referidos;

- No **Capítulo 3**, é delineado o processo para a realização de um qualquer estudo deste tipo, sendo as principais etapas: a identificação da necessidade da sua criação, a especificação das questões de investigação e o desenvolvimento do protocolo (Secção 3.1), seguido da pesquisa, seleção, extração e sumarização de informação dos estudos primários (Secção 3.2) e, por fim, a definição da estratégia de divulgação do EMS e a escrita do documento (Secção 3.3);

- No **Capítulo 4**, é detalhado todo o trabalho realizado ao longo da dissertação: o processo de revisão (Secção 4.1), a discussão dos resultados obtidos (Secção 4.2), a avaliação do EMS por profissionais da área (Secção 4.3) e possíveis ameaças à validade do estudo (Secção 4.4);

- Por fim, no **Capítulo 5**, são descritas as conclusões deste trabalho, a sua principal contribuição (Secção 5.1), algumas limitações, que podem ser comuns a qualquer estudo deste tipo (Secção 5.2) e sugestões para trabalho futuro (Secção 5.3).

Capítulo 2

Enquadramento

2.1 Computação de Alto Desempenho, no processamento de *Big Data*

Para que se conheçam as linguagens existentes para este tipo de computação vai ser, na atual Secção, estudado o conceito da Computação de Alto Desempenho, no processamento de *Big Data*.

Big Data tornou-se uma das expressões mais utilizadas dos nossos tempos. Investigadores e profissionais dedicaram-se a desenhar ferramentas e técnicas inovadoras para acompanhar a rápida evolução e a crescente complexidade de grandes problemas científicos. *Big Data* foi definido como o modelo 3Vs [Bak12; BL12; DM+15; SS13]:

> *Big Data são conjuntos de informação de elevado Volume, Velocidade e/ou Variedade que exigem formas inovadoras e económicas de processamento, que permitem uma melhor percepção, tomada de decisões e automação de processos.*

Big Data e a respetiva análise são o centro da ciência moderna e dos negócios. *Deep Learning* (um ramo de *Machine Learning* baseado num conjunto de algoritmos que tentam modelar abstrações de alto nível de dados) é uma ferramenta útil para analisar e aprender a partir de *Big Data*. Estes dados podem ser não rotulados e não estruturados, dificultando a sua análise [Vie+17]. Os **cuidados a ter com a saúde**, analisando padrões de doenças e registos médicos, e a **inovação de novos produtos e serviços** são exemplos de utilização destes dados [How; SS13]. Compreendendo melhor o primeiro exemplo, conhecendo a sequência de nucleótidos no ADN humano é, por si só, de pouca utilização, mas perceber como esta sequência afeta o desenvolvimento e como poderá causar doenças é uma descoberta muito vantajosa [Pac11]. Tendo em conta o segundo caso, as empresas

têm a capacidade de recolher informação de fontes como as aplicações móveis, os *websites* e as redes sociais, conseguindo, após estudar esses dados, utilizá-los para a criação de novos produtos. No entanto, esta recolha e análise não é uma tarefa fácil, até porque, habitualmente, se tratam de dados não estruturados. Em 2012, entre outras estatísticas, concluiu-se que: a *Google* monitoriza 7,2 biliões de páginas e processa 20 *petabytes* de dados por dia, traduzindo, ainda, 66 idiomas; são criados cerca de 571 novos *websites* a cada minuto [SS13].

Existem três principais **componentes que caraterizam *Big Data*** [SS13], sendo estes:

- O enorme **volume** de dados existente [Mad12; Zik+11];
- A **velocidade**, por alguns processos serem limitados pelo tempo [Mad12; Zik+11];
- A grande **variedade** de fontes de onde provêm estes dados. Estes dados, geralmente, podem ser de um de três tipos: **estruturados** (inseridos num conjunto já definido e facilmente ordenados), **não estruturados** (aleatórios e de difícil análise), **semi-estruturados** (não estão em conformidade com campos fixos mas contêm *tags* para separar os diversos elementos) [SS12; Zik+11].

Independentemente do tamanho dos dados, deparamo-nos com dois grandes passos para os analisar, sendo que, em cada um deles, existe trabalho para ser feito e várias dificuldades a superar [LJ12]:

1. **Aquisição dos Dados** → uma grande parte dos dados produzidos pelas fontes de dados, como as redes sociais, não têm interesse, devendo ser filtrados. Um dos desafios é definir esses filtros de forma a que não seja descartada informação útil; outro consiste na descrição de que dados deverão ser guardados e como (tendo em atenção, p.e., os duplicados);
2. **Processo de Extração de Informação** → neste momento, é extraída a informação necessária das fontes e expressa de forma estruturada.

O processamento de *Big Data* requer uma grande quantidade de recursos de armazenamento e computação. Durante muitos anos, os processadores foram sendo, de ano para ano, cada vez mais rápidos mas, desde 2002, a taxa de melhoria do desempenho em processadores simples diminuiu bruscamente (de 50% para 20%). Para um aumento do poder computacional dos processadores, foram utilizados múltiplos *cores*, num único *chip* [Pac11].

O objetivo da CAD consiste em estudar as metodologias e técnicas que permitem a exploração de arquiteturas de hardware com múltiplos cores (utilizando-os paralelamente), com a finalidade de tornar mais eficiente, fiável e rápida a execução de programas que necessitam de recursos computacionais elevados [LS09; Pac11].

O desafio da computação científica, no contexto de *Big Data*, é a necessidade de combinar o desenvolvimento de *Software* para a CAD com a gestão e análise de *Big Data* [Bro+11; SS13].

Os sistemas de computação paralela, atualmente, podem incluir processadores *multi-core* e *many-core*. Os processadores ***multi-core*** possuem dois ou mais *cores* e são adequados para tarefas gerais. Os ***many-core*** compreendem um número maior de *cores* e têm um bom desempenho em tarefas específicas [Mem+17]. O *many-core* mais popular, atualmente, disponível são as *Graphics Processing Units (GPU)*.

Esta alteração efetuada para aumentar o poder dos processadores teve uma grande consequência para os desenvolvedores de *Software*, pois, adicionando simplesmente mais *cores* não iria melhorar o desempenho dos programas em série [Pac11]. Para uma melhor compreensão, na Figura 2.1, é apresentado um gráfico que demonstra o desempenho do programa, tendo em conta a porção de código paralelizável e a quantidade de processadores a ser utilizada.

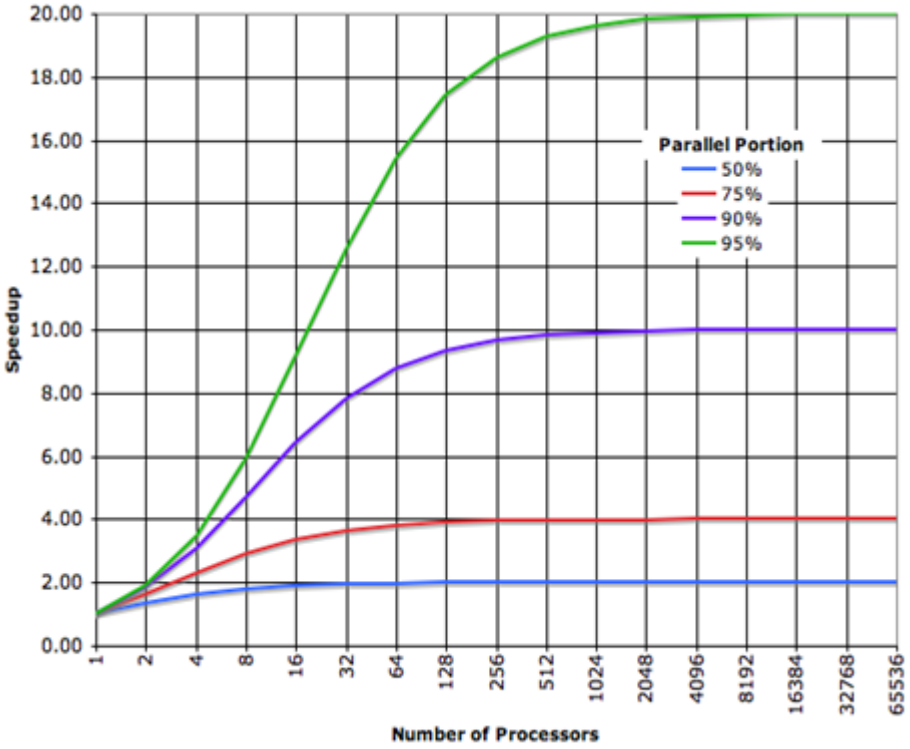

Figura 2.1: Desempenho de um programa (adaptado de [Amd])

Como demonstrado, quanto maior for a porção de código paralelizável, melhor será o desempenho dos programas. Como expectável, dentro de um certo limite, este aumento do *speedup* no desempenho também se irá verificar à medida que se vai recorrendo a mais processadores. Para resolver o problema da não melhoria do desempenho dos programas em série, é necessário tentar reescrever estes programas (pelo menos, parcialmente), com

a finalidade de aumentar a porção de código paralelizável. Por essa razão, os desenvolvedores de *Software* terão de aprender a escrever programas paralelos.

Devido às suas características diferentes, os engenheiros geralmente combinam processadores *multi-core* e *many-core* para conseguirem alcançar um alto desempenho e eficiência energética. Alguns esforços foram feitos para fornecer aos programadores ferramentas adequadas para o mapeamento de computações paralelas de dados, tanto em *multi-cores* como em *GPU*. Dois grandes problemas relacionados com esses ambientes e ferramentas de programação são a programabilidade e a eficiência. A **programabilidade** é, frequentemente, prejudicada pelo nível de abstração fornecido ao programador. A **eficiência**, em geral, sofre de peculiaridades relacionadas com a exploração efetiva da hierarquia de memória [Ald+17]. Como consequência, existem duas necessidades distintas: por um lado, são necessários mecanismos cada vez mais eficientes que suportem o acesso concorrente correto a estruturas de dados de memória partilhada; por outro, há a necessidade de ambientes de programação de alto nível, capazes de esconder as dificuldades relacionadas com o uso correto e eficiente de objetos de memória partilhada, aumentando o nível de abstração fornecido aos programadores das aplicações. A **programação paralela de alto nível** (do inglês *High-level parallel programming*) é um tópico de pesquisa destinado a promover metodologias de programação paralela que fornecem ao programador abstrações de alto nível para o desenvolvimento de *Software* paralelo complexo [DS+17].

Para lidar com o comprimento ilimitado das *streams* de dados, existem técnicas para aplicar, repetidamente, o processamento apenas nos tuplos mais recentes. Isso é permitido pela denominada **abordagem de processamento de janela deslizante** (do inglês *sliding window processing approach*), uma técnica para consultas (*queries*) que atualiza os seus resultados continuamente, à medida que novos dados chegam. Uma janela é um conjunto limitado dos tuplos mais recentes, cujo conteúdo é determinado de acordo com as opções tomadas pelo programador (modelos baseados em contadores, tempo ou híbridos) [Men+17].

Big Data gerou uma nova indústria de arquiteturas de suporte, como o *MapReduce*. O *MapReduce* é um modelo de programação, para a computação distribuída, que divide problemas complexos de *Big Data* em pequenas unidades de trabalho e processa-as em paralelo. O *MapReduce* pode ser dividido em duas etapas [SS13]:

- ***Map*** → O *input* é dividido em vários sub-problemas (pares chave-valor). De seguida, esses pares são processados e o resultado é armazenado em pares intermédios chave-valor;

- ***Reduce*** → Todos os pares intermédios são associados com outros pares com a mesma chave. Esses dados são, novamente, processados, gerando um único *output*.

Para uma melhor compreensão deste modelo, um exemplo da sua implementação é a contagem de palavras num documento (Figura 2.2): inicialmente, este ficheiro de *input* é dividido em vários sub-problemas, que são mapeados em pares chave-valor, onde a chave é a própria palavra e o valor é o número "1" (a palavra apareceu uma vez) - ***Map***; seguidamente, os pares intermédios com a mesma chave vão ser agrupados e, a este ponto, o valor destes pares será uma lista com os "1" respetivos - ***Reduce***; por fim, o *output* deste problema será constituído por todos os pares chave-valor, sendo a chave a palavra encontrada e o valor o número total de elementos da lista de "1" [LD10].

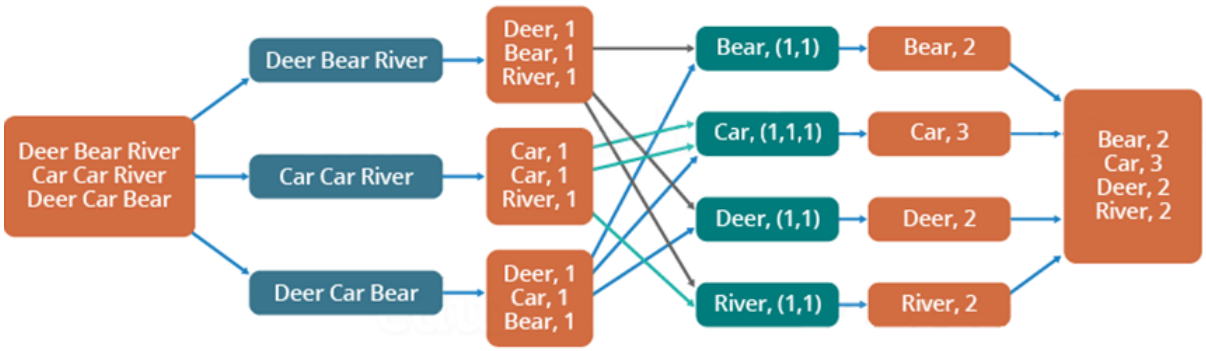

Figura 2.2: Contagem de palavras num ficheiro de *input* - Exemplo de implementação do modelo de programação *MapReduce* [Map]

O *Hadoop* é um *framework* que permite o processamento distribuído de grandes conjuntos de dados em *clusters* de computadores. O *Hadoop* foi desenhado/projetado para escalar de servidores únicos para milhares de máquinas, cada uma oferecendo computação e armazenamento locais. Em vez de confiar no *hardware* para fornecer alta disponibilidade, são detectadas e reparadas as falhas na camada da aplicação, entregando, assim, um serviço altamente disponível num *cluster* de computadores. Cada um destes pode estar sujeito a falhas [Had; Bar+13; Mar13; SS13].

Nos últimos anos, as ferramentas de processamento de *streams* (do inglês *Stream Processing Engines*) tornaram-se componentes principais no ramo de *Big Data* [Men+17]. Exemplos conhecidos destas ferramentas são *Storm* e *Spark*. Com estas ferramentas, os dados são consumidos e tratados, reparticionando as *streams* em várias coleções de objetos, que podem ser recriadas em caso de perda de uma partição [Sto; Mis+17; Zah+10].

2.2 Linguagens de Domínio Específico e Linguagens de Propósito Geral

Na presente Secção, vão ser definidos e comparados os dois tipos de linguagens existentes, LDE e LPG, tendo em conta que esta dissertação se baseia nas linguagens encontradas para a Computação de Alto Desempenho, utilizadas no processamento de *Big Data*, e as categorias atuais das mesmas são um tópico importante das cinco questões de investigação formuladas.

Em todos os ramos da Ciência e da Engenharia existe uma distinção entre abordagens **genéricas**, que fornecem uma solução geral, e **específicas**, caraterizadas por oferecer uma solução melhor para um conjunto mais limitado de problemas.

Uma Linguagem de Domínio Específico (LDE) é adaptada a um domínio de aplicação específico e oferece anotações e abstrações apropriadas [Kos+10; Kos+16; VD+00].

Na Figura 2.3, encontra-se um exemplo de código em *DOT*, uma linguagem de descrição de grafos [GN00], que cria um grafo não direcionado e outro direcionado, respetivamente. Um outro exemplo de LDE é *Csound*, também conhecida como LDE de áudio.

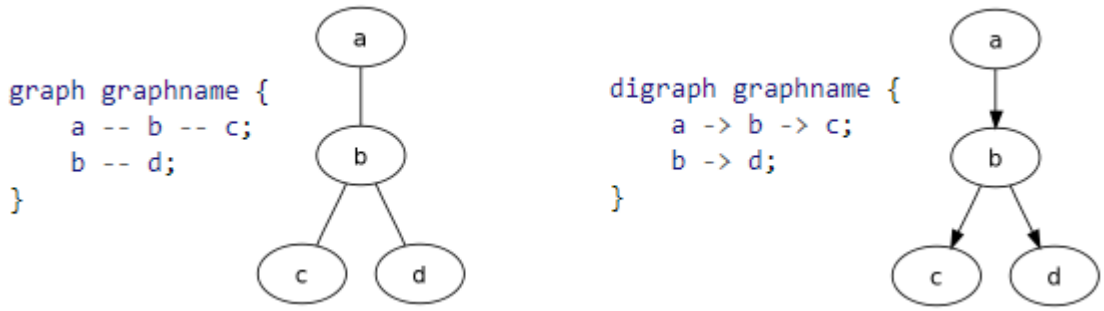

Figura 2.3: Exemplo de código escrito na linguagem *DOT* retirado de [GN00]

Uma Linguagem de Propósito Geral (LPG) é uma linguagem de programação projetada para ser utilizada na escrita de Software numa ampla variedade de domínios de aplicação [Kos+10].

No Apêndice I, encontra-se um exemplo de código em *Java*, que implementa um grafo direcionado ou não direcionado, dependendo do valor da variável booleana *directed*. Este código foi retirado de [Jav].

As linguagens de programação comuns como *C*, *C#*, *C++*, *Java* e *JavaScript* são exemplos de LPG.

Na Tabela 2.1 é feita uma comparação relativamente aos dois tipos de linguagens definidos [Kos+10; Kos+12]:

Tabela 2.1: Comparação entre LDE e LPG

Linguagens de Domínio Específico	**Linguagens de Propósito Geral**
Especializadas num problema de domínio específico, fornecem uma notação próxima ao domínio da aplicação e são baseadas nos conceitos e recursos desse domínio	São gerais, o que resulta num suporte fraco para a notação específica do domínio em questão
Para um domínio em questão são, em princípio, mais fáceis de utilizar[1]	Não sendo especializadas em nenhum domínio, podem ser boas em vários
Existe a possibilidade de integração de especialistas em etapas posteriores do ciclo de vida de desenvolvimento do *Software*	Os especialistas do domínio podem não ter a experiência necessária em programação para conseguir tirar partido de uma LPG
Estas linguagens fornecem documentação própria, podendo simplificar a manutenção do *Software*. No entanto, por motivos de escala da comunidade que as utiliza, é mais provável encontrar uma LPG bem documentada do que uma LDE	Nem sempre existe documentação disponível, tornando a tarefa de manutenção do *Software* mais difícil mas, em princípio, será mais fácil encontrar documentação destas linguagens do que das LDE

[1] Apesar de, na comparação de LDE com LPG, se assumir que as LDE são mais fáceis de utilizar, isso não é necessariamente verdade [Gab+10].

2.3 Engenharia de *Software*

Nesta Secção, vão ser apresentadas as duas áreas relevantes da Engenharia de *Software* para se conseguir compreender a elaboração de um EMS e todas as etapas envolvidas nesse processo: Engenharia de *Software* Experimental e a Engenharia de *Software* Baseada em Evidências.

A Engenharia de Software (ES) (do inglês Software Engineering) é a área da Engenharia que se preocupa com todos os aspetos da produção de Software, desde as fases iniciais de especificação até à sua manutenção depois de ter sido utilizado [PA09; PM14; Som15].

A ES preocupa-se com teorias, métodos e ferramentas para o desenvolvimento profissional de *Software*.

Os autores da publicação [Bas+08], com a realização de experiências, observaram que as tecnologias da ES que não tenham em conta as necessidades dos profissionais da CAD não são adoptadas. Exemplos destas situações são:

- As **tecnologias orientadas a objetos (OO)** estão firmemente enraizadas na comunidade da ES. No entanto, na comunidade da CAD as linguagens *C* e *Fortran* ainda dominam, embora *C++*, *Java* e *Python* também sejam exploradas. Uma razão para este acontecimento pode ser o facto das linguagens OO, como *C++*, estarem a evoluir mais rapidamente do que *C* e *Fortran*, nos últimos anos, tornando-se escolhas mais arriscadas;

- Os cientistas ainda precisam de ser convencidos de que a **reutilização de *frameworks* existentes** lhes poupará mais esforço do que construir as suas próprias, desde o início. Estes *frameworks* fornecem aos programadores um nível mais elevado de abstração, mas ao custo de adoptar a perspectiva do *framework* sobre como estruturar o código. Na opinião destes especialistas, encaixar o seu problema num desses *frameworks* exigirá mais esforço do que construir os seus próprios;

- Não são utilizados ***Integrated Development Environments (IDE)***, porque estes não se encaixam bem no fluxo de trabalho típico de um cientista que executa um código num sistema de CAD. Por exemplo, os *IDE* não suportam *debugging* para máquinas paralelas.

2.3.1 Engenharia de *Software* Experimental

A Engenharia de Software Experimental (ESE)[2] (do inglês Empirical Software Engineering) é uma área de pesquisa da ES que utiliza métodos empíricos como a análise de informação baseada em experiências e observações sobre um determinado tema [Mal16].

Os estudos empíricos podem fornecer evidências poderosas para testar hipóteses dadas [Mal16]. A ES deve adotar métodos empíricos que ajudarão a planear, avaliar, estimar, monitorizar, controlar, prever, gerir e melhorar a forma como os produtos de *Software* são produzidos [Mal16].

A sua utilização tem como **principais vantagens** [Mal16]:

- Apoiar o suporte de conceitos teóricos;
- Ajudar a verificar, validar e melhorar, recorrendo ao método científico, ferramentas e técnicas no desenvolvimento de *Software*;
- Permitir estabelecer parâmetros de qualidade nas organizações de *Software*.

Existem três tipos diferentes de análise que podem ser realizadas:

- **Qualitativa** → refere-se a um conjunto de técnicas de investigação como a observação e as entrevistas, não-estruturadas, isto é, onde o entrevistador desenvolve as conversas de uma forma espontânea, sem a definição prévia de questões, e semi-estruturadas, que obedecem a um pequeno guião [Infa; CG02; Sea99];
- **Quantitativa** → aplica métodos quantitativos, como as estatísticas, para derivar conclusões. Esta baseia-se numa medição controlada [Infb; CG02];
- **Semi-Quantitativa** → sendo um misto dos dois anteriores, associa observação e alguma medição.

Segundo *Petticrew* e *Roberts* [PR05], alguns estudos são melhores do que outros na abordagem de diferentes tipos de questões de investigação (p.e. os estudos qualitativos são mais apropriados do que as experiências aleatórias para avaliar se os profissionais encontraram uma nova tecnologia apropriada para o tipo de aplicações que precisam de construir [KC07]). Deste modo, ao realizar um estudo secundário, se queremos restringir-nos a estudos de um tipo específico, devemos optar por aquele que é mais adequado para responder às nossas questões.

[2] O termo internacionalmente mais utilizado é "*Empirical Software Engineering*". Em português, a expressão preferida é "Engenharia de *Software* Experimental" porque o termo "Empírico" tem uma conotação negativa, ao contrário do que acontece em inglês.

Ao longo da literatura existente, podemos deparar-nos com: experiências repetidas, que não são avaliadas como contribuições importantes para a pesquisa; teorias não testáveis, não sendo possível determinar se a teoria é válida ou não [Vot+95]. Como tal, para a criação de um EMS, outro tipo de estudo secundário, ou uma revisão terciária, não deverão ser incluídos estes estudos, para que: os resultados não sejam influenciados, devendo optar-se pelo artigo mais completo, no caso em que se verifica a repetição de experiências; a credibilidade do nosso EMS não seja reduzida, no caso da teoria poder não ser válida.

2.3.2 Engenharia de *Software* Baseada em Evidências

Em muitos casos, o *Software* é construído utilizando tecnologias, métodos ou ferramentas, das quais não há evidências suficientes que confirmem a sua adequação ao contexto, limites da tecnologia, qualidade, custos e riscos inerentes à utilização. Sendo assim, foi adaptado de outras áreas, como a Medicina, o conceito de Engenharia de Software Baseada em Evidências (ESBE) (do inglês *Evidence-Based Software Engineering*) [Dyb+05; KC07; Kit+04].

Na ES, a evidência é definida como uma síntese de estudos científicos sobre um tópico específico ou uma questão de pesquisa [Kit+09; Kit+04].

Esses estudos podem ser classificados em três tipos distintos [KC07], sendo estes:

- **Estudos Primários** (p.e. Estudos de Caso, *Surveys*) → Um estudo empírico (descrito na Subsecção 2.3.1) que investiga uma questão de investigação específica;
- **Estudos Secundários** (p.e. Estudo de Mapeamento Sistemático (EMS), Revisão Sistemática de Literatura) → Um estudo que revê todos os estudos primários relacionados com uma questão de pesquisa específica, com o objetivo de sintetizar essas evidências;
- **Estudos Terciários** (também chamado de Revisão Terciária) → Revisão de estudos secundários relacionados com uma mesma questão de investigação. Estes são criados com a finalidade de responder a questões de pesquisa mais amplas [Kit+10].

A finalidade da ESBE é melhorar a tomada de decisões relacionadas com o desenvolvimento e manutenção de Software, como ajudar a decidir quais são as tecnologias apropriadas num dado contexto em detrimento das restantes, recolhendo e avaliando a melhor evidência [Jor+05; Kit+02; Kit+04].

A ESBE envolve cinco passos principais [Dyb+05; Jor+05; Kit+04; Sac+00], descritos de seguida:

1. Uma vez identificado o problema, é necessário **especificar uma questão de investigação**. É essencial converter um problema numa questão que seja específica, mas não demasiado, para que consiga ser respondida;

2. **Procurar a melhor evidência que responda à questão proposta**. Para tal, é fundamental selecionar um recurso de informação apropriado e executar uma estratégia de pesquisa. Exemplos de fontes dessas evidências são as revistas científicas e as conferências;

3. Para avaliar se a pesquisa é de boa qualidade e os seus resultados são aplicáveis à prática, terá de se **avaliar a evidência encontrada**;

4. **Aplicar a evidência**. Um desenvolvedor de *Software* deve envolver-se ativamente num processo de aprendizagem, combinando a evidência em questão com a experiência e circunstâncias do cliente, de modo a tomar decisões acerca da prática;

5. Perguntar-se o quão bem estão a ser integradas as evidências com a experiência, requisitos do cliente e o conhecimento das circunstâncias específicas. Uma maneira simples de uma equipa descobrir os seus sucessos e falhas é através de reuniões curtas destinadas a **avaliar o seu desempenho**.

É sabido que a ESBE oferece uma grande variedade de **benefícios** aos profissionais na área do *Software* e aos seus clientes e utilizadores. Em particular, a utilização de técnicas apoiadas por evidências deve melhorar a qualidade dos sistemas e assegurar aos grupos de partes interessadas, como é o caso dos clientes, que os profissionais estão a recorrer às melhores práticas [Kit+02; Kit+04]. No entanto, a escolha das tecnologias deverá ser influenciada por factores como a experiência dos desenvolvedores de *Software*, os requisitos dos clientes e as restrições do projeto em questão.

2.4 Estudo de Mapeamento Sistemático

Na presente Secção, é definido um Estudo de Mapeamento Sistemático (EMS) e o outro tipo comum de estudo secundário existente (Revisão Sistemática de Literatura). Por fim, são comparados esses dois tipos de estudos.

> *Um Estudo de Mapeamento Sistemático (EMS) é projetado para fornecer uma visão ampla, construída sobre questões gerais, de uma área de pesquisa para estabelecer se existem evidências de pesquisa sobre um determinado tópico e fornecer uma indicação dessas evidências [KC07; Pet+08a; Pet+08b].*

Para além do EMS, existe um outro tipo de revisão que o complementa, sendo este:

- **Revisão Sistemática de Literatura** (RSL, do inglês *Systematic Literature Review*) → uma forma de identificação, avaliação e interpretação de todas as pesquisas disponíveis relevantes para uma determinada questão de pesquisa, área ou fenómeno de interesse [KC07; Pet+08a; Pet+08b].

Um EMS pode ser conduzido primeiro, para obter uma visão geral do tópico e, seguidamente, os tópicos específicos podem ser investigados, recorrendo a uma RSL.

As principais diferenças entre um EMS e uma RSL são [KC07; Pet+08b]:

- Um EMS, geralmente, dá resposta a questões de investigação mais amplas tratando-se, muitas vezes, de mais do que uma questão;
- Os termos de pesquisa para um EMS são menos focados do que para uma RSL e, usualmente, retornam uma maior quantidade de estudos;
- O processo de extração de informação de um EMS é mais amplo do que o de uma RSL. O objetivo desta etapa é classificar os artigos com detalhes suficientes para responder às questões propostas e identificar artigos para análises posteriores;
- A fase de análise de um EMS consiste em resumir os dados para responder às questões de investigação apresentadas, não incluindo técnicas de análise aprofundada, como a meta-análise[3];
- A divulgação dos resultados de um EMS tem como objetivo influenciar futuras direções de pesquisa.

[3] A meta-análise é uma forma de estudo secundário, cuja síntese da pesquisa é baseada em métodos estatísticos quantitativos.

2.5 Sumário

O objetivo da CAD consiste em estudar as metodologias e técnicas que permitem a exploração de arquiteturas de *hardware* com múltiplos *cores* (utilizando-os paralelamente), com a finalidade de tornar mais eficiente, fiável e rápida a execução de programas que necessitam de recursos computacionais elevados [Pac11]. Um dos principais exemplos de utilização das técnicas e metodologias estudadas pela área mencionada é *Big Data* [AV15], que consiste na gestão de dados armazenados, de um elevado volume, variedade e/ou velocidade [Bak12; BL12; DM+15; SS13].

A ES é a área da Engenharia que se preocupa com todos os aspetos da produção de *Software* desde as fases iniciais da especificação do sistema à sua manutenção depois de ter sido utilizado [PA09; PM14; Som15]. A ESE é a área de pesquisa da ES que utiliza métodos empíricos como a análise de informação baseada em experiências e observações sobre um determinado tema [Mal16]. A ESBE pretende aplicar uma abordagem baseada em evidências para a prática da ES.

Os estudos individuais que contribuem para um EMS são denominados estudos primários. Um EMS é um estudo projetado para fornecer uma visão ampla, construída sobre questões gerais, de uma área de pesquisa para estabelecer se existem evidências de pesquisa sobre um determinado tópico e fornecer uma indicação dessas evidências [KC07; Pet+08a; Pet+08b]. Um estudo terciário é uma revisão de estudos secundários relacionados a uma mesma questão de pesquisa e deve ser realizado num tópico onde já existam estudos em número suficiente.

Capítulo 3

Processo de Revisão

3.1 Planeamento do EMS

Existem três grandes momentos no ciclo de vida do desenvolvimento de um EMS sendo, na Estrutura Analítica de Projetos (EAP, do inglês *Work Breakdown Structure*), afigurada em 3.1, denominados de: Planeamento, Condução e Elaboração [KC07; Kit+09; Mal16; Pet+08a].

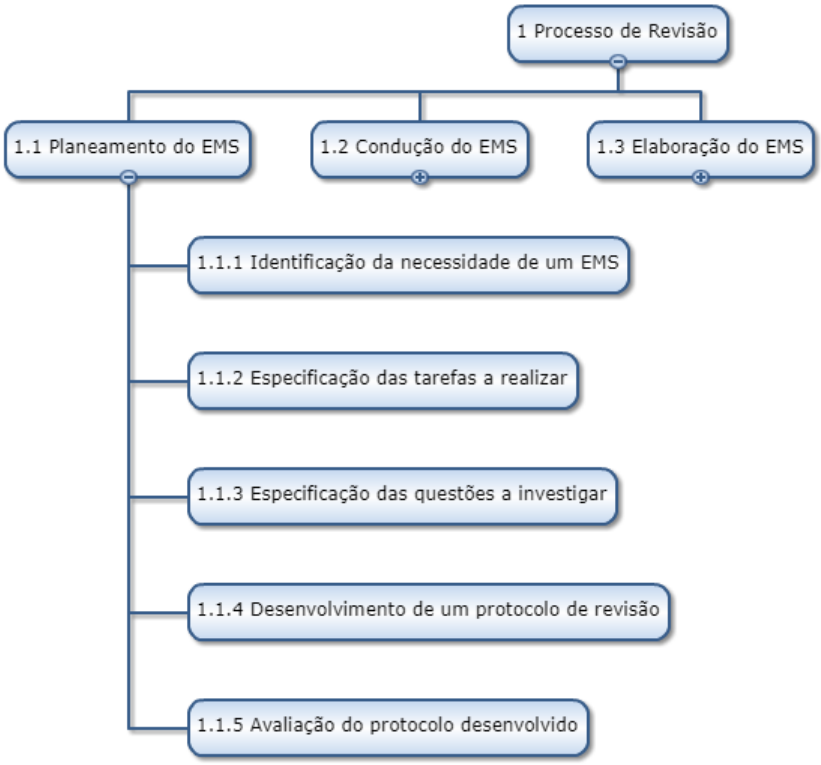

Figura 3.1: EAP do processo de Planeamento de um EMS

As etapas descritas ao longo do capítulo podem aparecer de modo sequencial mas algumas envolvem iterações, mais especificamente, muitas das atividades são iniciadas durante a fase de desenvolvimento do protocolo e refinadas posteriormente [KC07]. Exemplos destes casos são:

- A seleção de estudos primários é baseada nos critérios de inclusão e exclusão definidos no protocolo, mas estes poderão ser retificados;

- Os formulários de extração de dados, inicialmente preparados durante a construção do protocolo, poderão ser alterados;

- Os métodos de síntese de informação definidos no protocolo poderão vir a ser alterados.

Antes de realizar um EMS, a sua necessidade tem de ser confirmada. As atividades de planeamento do EMS mais importantes são a **definição das questões de pesquisa** a ser abordadas e a **produção de um protocolo de revisão**, onde são definidos os procedimentos básicos [KC07; Kit+09; Mal16; RD09].

- A **necessidade de um EMS** decorre da importância de resumir, de forma imparcial, todas as informações existentes sobre algum tópico. Antes de realizar um EMS, deverá ser garantida a sua necessidade, devendo, em particular, identificar e rever quaisquer estudos secundários existentes sobre esse assunto;

- Por vezes, uma organização requer informações sobre um tópico específico, mas não tem tempo ou experiência para realizar um EMS. Nesses casos, é pedido a investigadores que o realizem. Quando isso ocorre, a organização deverá produzir um documento, **especificando as tarefas a realizar**, que conterá informações como: as perguntas e métodos de revisão, a calendarização do projeto e a estratégia de divulgação. A escrita deste documento é opcional, sendo que, caso não seja efetuada, a estratégia de divulgação deverá estar definida no protocolo de revisão;

- A **formulação das questões de pesquisa** é a etapa mais importante de qualquer estudo secundário, pois, são estas questões que conduzem toda a sua metodologia: o **processo de pesquisa** deve identificar os documentos que abordem as questões de investigação, o **processo de extração de informação** deve extrair os dados necessários para responder às perguntas propostas, o **processo de análise de dados** deve sintetizá-los de forma a que as perguntas consigam ser respondidas. A questão crítica em qualquer estudo deste tipo é formular as questões corretas, devendo estas ser significativas e importantes para todos os envolvidos (profissionais e investigadores). Recentemente, *Petticrew* e *Roberts* [PR05] sugeriram o uso dos critérios *PICOC* (População, Intervenção, Comparação, Resultados (*Outcomes*), Contexto), definidos em Engenharia de *Software*, para enquadrar as questões de pesquisa [KC07;

RD09]. Em áreas onde exista um elevado número de artigos sobre o tema, poderá ser definido um critério extra, o Desenho Experimental:

1. **População** → entre outros exemplos, poderá ser um papel específico, uma área de aplicação ou um conjunto de estudos;

2. **Intervenção** → metodologia/ferramenta/tecnologia de *Software* que aborda um problema específico, p.e. tecnologias para executar tarefas específicas, tais como a especificação de requisitos ou testes ao sistema;

3. **Comparação** → metodologia/ferramenta/tecnologia de *Software* com a qual a intervenção está a ser comparada. Por normalmente necessitarem de treino, caso sejam comparadas pessoas que as utilizam com pessoas que não as conhecem, o seu efeito poderá ser confundido com o efeito do treino;

4. **Resultados** → devem ser relacionados com factores de importância para os profissionais, como uma melhor confiabilidade e redução dos custos de produção. Em alguns casos, são exigidas intervenções que melhorem algum aspeto da produção de *Software* sem afetar outro p.e. uma maior confiabilidade sem o aumento de custo de produção;

5. **Contexto** → onde ocorre (p.e. escola, indústria), os participantes envolvidos no estudo (p.e. estudantes, profissionais) e as tarefas que estão a ser executadas (p.e. de pequena, grande escala);

6. **Desenho experimental** → em alguns tópicos, a natureza da questão pode sugerir que certos tipos de estudos são mais apropriados do que outros, restringindo-se os investigadores aos primeiros. No entanto, esta abordagem apenas pode ser adoptada caso se encontrem demasiados artigos, algo improvável na Engenharia de *Software*. Se apenas optarmos por uma amostra de artigos, numa pequena porção de estudos disponíveis, existe o risco da cobertura de artigos ser incompleta.

- O **desenvolvimento de um protocolo de revisão**, que descreve os métodos que serão utilizados para realizar um estudo secundário. Um protocolo pré-definido é necessário para reduzir a possibilidade de introduzir enviesamentos (mesmo que involuntários) na selecção e análise de estudos primários. Sem um protocolo, é possível que a seleção de artigos ou a análise possa ser conduzida pelas suas expetativas. Os componentes do nosso protocolo de revisão encontram-se especificados no Apêndice B;

- O protocolo desenvolvido é um elemento crítico de qualquer estudo secundário, devendo a sua consistência ser verificada. Para tal, deverá confirmar-se que: as *strings* de pesquisa são adequadamente derivadas das questões de pesquisa especificadas; os dados a serem extraídos darão resposta, corretamente, às questões referidas; o procedimento de análise de dados é apropriado para as responder.

3.2 Condução do EMS

Os passos para a Condução do EMS, resumidamente, para a pesquisa, recolha e sumarização da informação, estão exibidos na EAP, da Figura 3.2.

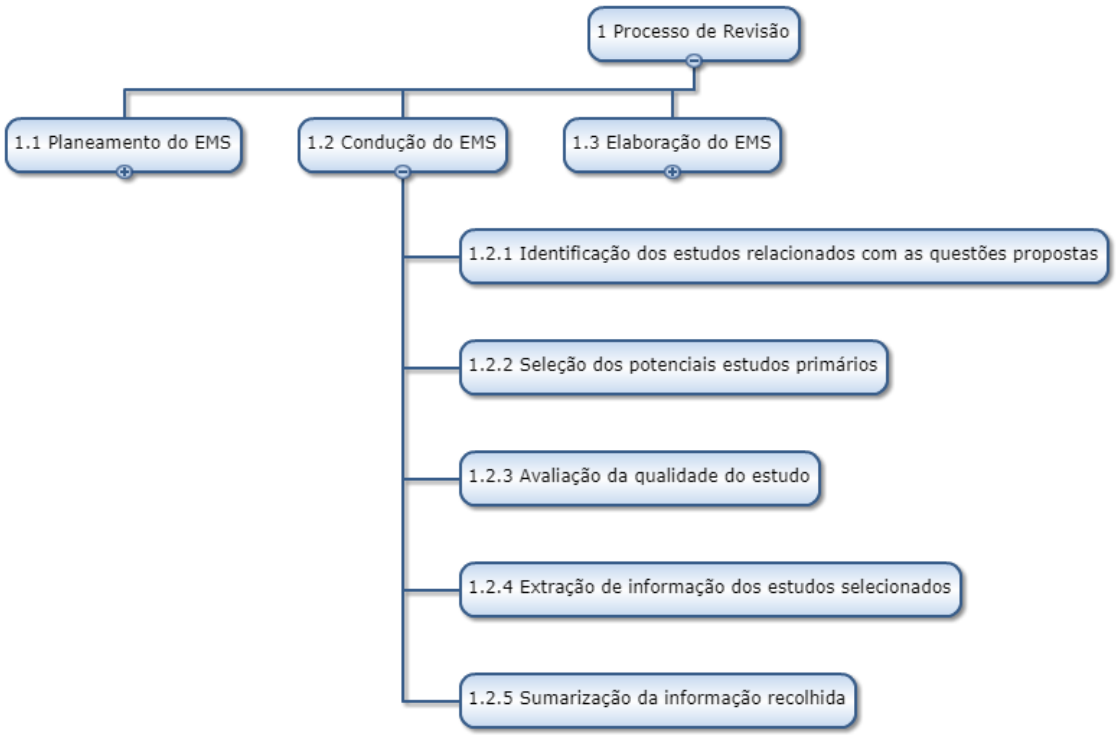

Figura 3.2: EAP do processo de Condução de um EMS

Uma vez chegado a um consenso relativamente ao protocolo desenvolvido, procederemos com a condução do EMS [KC07; Kit+09; Mal16; RD09].

- A deteção de revistas científicas e conferências relevantes, tendo em conta o tópico em questão, e posterior geração de *strings* de busca adaptadas para as diferentes bibliotecas digitais existentes (terá de ser tido em conta o problema de publicações tendenciosas, pelo facto dos resultados positivos serem, normalmente, mais publicados que os negativos), de modo a **identificar os estudos relacionados com as questões propostas**;

- Uma vez obtidos os artigos relevantes, aplicar os critérios de seleção (critérios de inclusão e exclusão dos estudos) definidos pelos autores do EMS, baseados p.e. na língua em que foram escritos, na sua área, autores e ano de publicação, **selecionando os potenciais documentos** a examinar e efetuando o respetivo registo. Aquando da incerteza de incluir ou excluir um estudo, esta decisão deverá ser confirmada por um outro investigador da equipa. Para minimizar o risco de exclusão acidental de estudos primários relevantes, uma técnica possível é, para uma amostra desses estudos, haver uma segunda pessoa a classificá-los de forma independente. Em caso de concordância, existe maior confiança na decisão. Em caso de discordância, deve haver uma discussão entre os revisores discordantes, de modo a que daí resulte uma classificação consensual. Esta técnica tem o benefício adicional de permitir estimar quantos estudos levariam a uma discordância entre revisores, além de facilitar decisões mais consistentes entre os diversos revisores, pela clarificação de critérios que o processo de consensualização permite. Finalmente, em caso de dúvida, os artigos devem ser mantidos para análise mais detalhada, podendo vir a ser excluídos mais tarde se, na tentativa de extração de informação, se concluir que seria a melhor resolução;

- Como forma de **validação do processo de revisão**, influenciando a qualidade do estudo, segundo a *CRD Database of Abstracts of Reviews of Effects* (*DARE*) [KC07; RD09], deverão ser respondidas quatro questões fundamentais, sendo estas:

  1. Os critérios de inclusão e exclusão dos estudos estão descritos e são apropriados?
  2. A pesquisa literária cobriu todos os estudos relevantes?
  3. Os revisores avaliaram a qualidade do estudo?
  4. Os estudos primários foram adequadamente referidos?

- Numa próxima etapa, será produzido um formulário de **extração de dados** para recolher todas as informações necessárias. Além de incluir todas as perguntas necessárias para responder às questões propostas do EMS, deverão ser fornecidos dados como: o nome do revisor; título; autores; outros detalhes da publicação. Alguns documentos deverão ser analisados por mais do que um revisor (p.e. uma amostra aleatória de estudos primários), para que a consistência dos resultados possa ser avaliada. Ao agregar informação, não deverão existir múltiplas publicações dos mesmos dados, devendo, em caso desta ocorrência, ser utilizada aquela que for considerada a mais completa ou ser feita uma fusão dessas várias publicações;

- A **síntese de dados** envolve a compilação e o resumo dos resultados dos artigos incluídos. Dependendo do tipo de estudo (qualitativo ou quantitativo) tem-se associada uma síntese qualitativa ou quantitativa, respetivamente. Numa **síntese qualitativa**, as informações extraídas devem ser tabeladas de forma estruturada para ressaltar as semelhanças e diferenças entre os resultados do estudo. Numa **síntese quantitativa**, os resultados do estudo devem ser apresentados de forma comparável. O mecanismo mais comum para a apresentação de resultados quantitativos é um *forest plot*, onde é apresentada a média e a variância da diferença para cada estudo.

3.3 Elaboração do EMS

As etapas da Elaboração do EMS são apresentadas na EAP, da Figura 3.3.

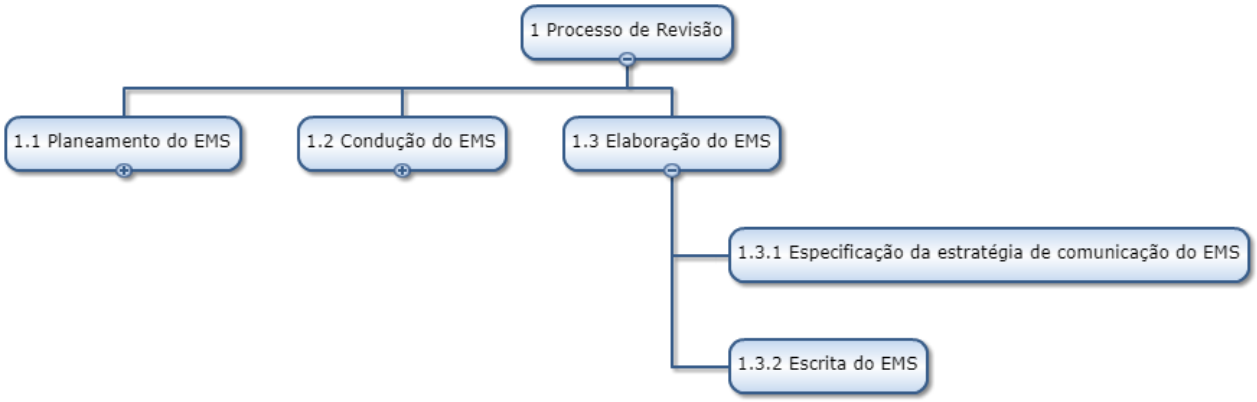

Figura 3.3: EAP do processo de Elaboração de um EMS

O último momento de um EMS envolve a redação dos resultados do estudo e a sua divulgação para as partes interessadas [KC07; Kit+09; Mal16; RD09].

- É importante **comunicar, eficazmente, os resultados de um EMS**. Se os resultados de um EMS pretendem influenciar os profissionais, é necessário recorrer a **formas de divulgação específicas**, em particular: revistas científicas, pequenos folhetos resumidos, *posters*, páginas *Web* ou comunicação direta aos corpos afetados;

- Por último, tem-se a fase de **formatação do EMS**. Normalmente, os EMS serão relatados em, pelo menos, dois formatos, mais especificamente, um relatório técnico ou secção de uma tese de mestrado/doutoramento e um artigo numa revista científica ou conferência.

Adicionalmente, no final do processo descrito, poderá existir uma fase de **avaliação do EMS**, realizada por investigadores que não a tenham escrito, isto é, uma avaliação independente.

3.4 Sumário

O processo de produção de um EMS é composto por três fases principais [KC07; Kit+09; Mal16; Pet+08a]:

- **Planeamento**, onde, no geral, se: identifica a necessidade de criação do EMS; especificam as questões a pesquisar; desenvolve um protocolo de revisão;

- **Condução**, sendo o momento em que são: especificados os estudos primários relacionados com as questões propostas e selecionados aqueles que devem ser incluídos no EMS; avaliada a qualidade do estudo efetuado; extraída a informação dos artigos listados; sumarizada a informação recolhida;

- **Elaboração**, que envolve a escrita dos resultados e a sua divulgação para as partes interessadas.

Capítulo 4

Trabalho Realizado

4.1 Processo de Revisão

Existem várias LPG e bibliotecas integradas nestas linguagens, mas é sabido que são difíceis de utilizar por parte dos profissionais de domínio que, tal como vimos anteriormente, não têm grandes competências em programação. De modo a tentar colmatar este problema, o foco do nosso estudo é identificar quais são as alternativas existentes a este tipo de linguagens, que lhes facilitem a tarefa de codificação.

A presente Secção descreve a realização de um EMS que procura dar uma visão panorâmica das linguagens existentes para a CAD, com apontadores para estudos mais especializados, referindo as várias linguagens identificadas, através da nossa pesquisa, e os estudos primários a elas associados. Nas respostas dadas às questões de investigação, foram retiradas conclusões sobre as caraterísticas gerais dessas linguagens.

A metodologia utilizada neste EMS (Figura 4.1) é organizada em seis passos sucessivos, propostos em [KC07; Kit+09] e listados no protocolo de revisão desenvolvido (Apêndice B). A delineação de todos os passos concluídos é descrita ao longo da Secção 4.1.

1. **Questões de Investigação**, com o objetivo de formular as questões de pesquisa a que o EMS pretende responder;

2. **Processo de Pesquisa**, que visa detetar o maior número de estudos primários relacionados com as questões de investigação propostas;

3. **Seleção de Estudos**, que filtra falsos positivos, recorrendo a um processo de seleção desenvolvido;

4. **Validação do Processo de Revisão**, respondendo a quatro questões fundamentais, referidas em [KC07; Kit+09];

5. **Processo de Extração de Informação**, onde, com base nos estudos selecionados, são dadas respostas às questões de investigação;

6. **Sumarização da Informação recolhida**.

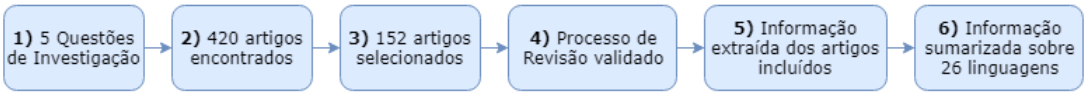

Figura 4.1: Metodologia utilizada neste EMS

O estudo foi conduzido de forma distribuída, com participantes de vários países europeus, pertencentes à rede *cHiPSet ICT COST Action* [Aboa], a colaborar, remotamente, durante todo o processo.

A minha função foi, não só a de me envolver, ativamente, em todos os períodos de desenvolvimento do EMS, mas, principalmente, a de coordenar todo o estudo, tendo a visão de como os dados vão ser recolhidos, analisados e sintetizados. Para concretizar o segundo ponto comuniquei, essencialmente, assincronamente com todos os colegas do grupo de trabalho onde fui inserida, distribuindo as tarefas por todos nós durante a seleção e análise dos estudos e pedindo, após sumarizar os dados e escrever a primeira versão do EMS num documento partilhado com todos os autores, que confirmassem o que tinha feito, melhorassem ou sugerissem melhorias. Terminadas as tarefas anteriores, solicitei a sua colaboração na escolha de qual seria a melhor conferência ou revista científica para comunicar o nosso estudo.

Realizei uma pesquisa nas bases de dados das plataformas electrónicas: *academia.edu*, *ACM Digital Library*, *Compendex*, *Elsevier Science Direct*, *Google Scholar*, *IEEE Xplore*, *Research Gate* e *Springer Link*, com a finalidade de confirmar a não existência de nenhum estudo que incidisse sobre as diversas linguagens utilizadas na CAD.

4.1.1 Questões de Investigação

O objetivo do EMS é responder a questões de pesquisa, dependendo desta tarefa a utilidade do estudo. A definição destas questões é uma etapa crucial do processo de revisão. Para enquadrar as questões de investigação, foram adoptados os critérios *PICOC* [1] (População, Intervenção, Comparação, Resultados (*Outcomes*), Contexto) [KC07], sendo os seus elementos definidos da seguinte forma:

- **População** → Composta pelos estudos primários encontrados sobre as Linguagens utilizadas para a CAD;

- **Intervenção** (metodologia/ferramenta/tecnologia de *Software* que aborda um problema específico) → Este EMS investiga os estudos sobre as Linguagens para a CAD, utilizadas no processamento de *Big Data*, descrevendo os seus detalhes, por exemplo, qual a categoria em que a linguagem se insere;

- **Comparação** (metodologia/ferramenta/tecnologia de *Software* com a qual a intervenção está a ser comparada) → Não aplicável a este caso;

- **Resultados** (relacionado com factores de importância para os profissionais) → Tecnologias, métodos e métricas que levam a um aumento da qualidade da solução: facilidade de configuração; usabilidade; ganhos de produtividade, tais como, uma linguagem de fácil aprendizagem ou utilização; ganhos de desempenho do produto, como, uma fácil manutenção e eficiência ao nível da memória;

- **Contexto** → Os participantes envolvidos neste estudo são investigadores e profissionais da área da CAD (membros da rede *cHiPSet ICT COST Action*).

Este EMS pretende responder a **cinco questões de investigação**, apresentadas na Tabela 4.1. Com este estudo, pretende-se descobrir:

1. Quais são as categorias das linguagens em utilização, por exemplo, se foram, maioritariamente, identificadas LDE ou LPG;

2. Qual é a natureza dessas linguagens, para que seja possível conhecê-las a vários níveis, como, as vantagens que proporcionam, as ferramentas de suporte existentes, o propósito e o seu tipo de representação preferencial;

3. Quais são os perfis de utilizador típicos para as linguagens, isto é, se habitualmente se tratam de utilizadores finais ou desenvolvedores para conceber novas soluções;

[1] Tendo em conta o referido em [KC07] e no Capítulo 3, na Engenharia de *Software*, a escassez de estudos primários é o obstáculo mais provável para um estudo secundário. De modo a não reduzir a quantidade de publicações, os investigadores não se restringem a estudos que considerem mais apropriados, não sendo definido, nestes critérios, o elemento "Desenho Experimental".

4. Quão eficazes são essas linguagens, sendo esta caraterística articulada em três aspetos (sucesso, ganhos de produtividade e vantagem em comparação com abordagens concorrentes);

5. Que tipos de artigos foram encontrados ao longo da nossa pesquisa.

As informações específicas relativas a cada uma das linguagens, individualmente, são apresentadas no Apêndice D.

Tabela 4.1: Questões de Investigação formuladas

ID da Questão	Questão de Investigação
QI 1	**Quais são as categorias das linguagens em utilização para a CAD?**
QI 1.1	Quais são as tendências atuais de pesquisa das linguagens para a CAD? (LDE, LPG, LDE embebidas em outras LDE, LDE embebidas em LPG)
QI 2	**Qual é a natureza das linguagens para a CAD?**
QI 2.1	Que tipo de linguagem é essa?
QI 2.2	Qual é o modelo de execução que está a ser utilizado? (*virtual execution environment*, *HPC libraries*, ...)
QI 2.3	Quais são as principais vantagens da linguagem? (desempenho, facilidade de configuração, usabilidade, ...)
QI 2.4	Qual é/são o/s domínio/s de aplicação da linguagem?
QI 2.5	Quais são os paradigmas subjacentes às linguagens? (declarativo, funcional, orientado a objetos, ...)
QI 2.6	Quais são os *execution stack requirements* para suportar os artefatos criados com essas linguagens? (*OS*, *Libraries*, ...)
QI 2.7	Qual é o suporte de ferramentas existente para a linguagem? (compiladores, simuladores, ...)
QI 2.8	Quais são as tecnologias utilizadas para criar o *tool suite* da linguagem? (*frameworks*, *XML based technology*, ...)
QI 2.9	A linguagem suporta *hardware* específico? A linguagem suporta *GPUs* ou arquiteturas *multi-core*?
QI 2.10	Qual é o propósito da linguagem? (simulação do problema, simulação da solução, formalização dos requisitos do problema, formalização dos requisitos da solução, implementação da solução, interpretação dos dados)

QI 2.11	Qual é o tipo de representação preferencial da linguagem? (textual e/ou diagramático)
QI 3	**Quais são os perfis de utilizador típicos para as linguagens?**
QI 3.1	Quais são os papéis dos utilizadores desta linguagem? (não é especificado, utilizador final, engenheiro a conceber uma solução para terceiros)
QI 3.2	Que tipo de conhecimento técnico é obrigatório? (*frameworks*, bibliotecas, *hardware*, conhecimento teórico, ...)
QI 4	**Quão eficazes são as linguagens?**
QI 4.1	O sucesso das linguagens é avaliado no artigo? (não é avaliado, é avaliado)
QI 4.2	Quais são os ganhos de produtividade trazidos pelas linguagens reportadas (aprendizagem fácil, memorização fácil, utilização fácil, ...) e que tipo de medição foi utilizado (quantitativa ou qualitativa)?
	Quais são os ganhos de desempenho trazidos pelas linguagens reportadas (eficiente ao nível da memória, de fácil manutenção, escalável, ...) e que tipo de medição foi utilizado (quantitativa ou qualitativa)?
QI 4.3	Existe uma comparação explícita com abordagens concorrentes?
	Existe uma comparação explícita da linguagem proposta com respeito a contextos/configurações distintas?
	Que tipo de comparação é realizada? (quantitativa, qualitativa ou ambas)
	Qual é a metodologia de comparação?
	Quais são as métricas utilizadas? (linhas de código, satisfação, tempo, ...)
QI 5	**Que tipos de artigos são publicados na área de modelos de programação para a CAD?**
QI 5.1	A publicação inclui autores da *COST CHiPSet*?
QI 5.2	Quais são as instituições envolvidas?
QI 5.3	Qual é o nome da conferência ou revista científica?
QI 5.4	Quem patrocina a pesquisa? (fundos públicos, fundos privados, ambos)
QI 5.5	Que tipo de pesquisa está a ser reportada? (estudo de caso, relatório de experiência, avaliação comparativa)

4.1.2 Processo de Pesquisa

Um objetivo importante para conduzir um estudo secundário é **detetar o maior número de estudos primários relacionados com as questões de investigação propostas**.

Foi adoptado um processo baseado em duas etapas principais. Primeiramente, foi utilizada uma única base de dados (a da biblioteca digital *Elsevier Science Direct*) e refinada a *query* de pesquisa, abrangendo todas as palavras-chave, de modo a ser detetado um grande conjunto de artigos de interesse para o EMS. Posteriormente, foram utilizadas *queries* similares para pesquisar num conjunto mais extenso de bases de dados. O processo encontra-se esboçado na Figura 4.2.

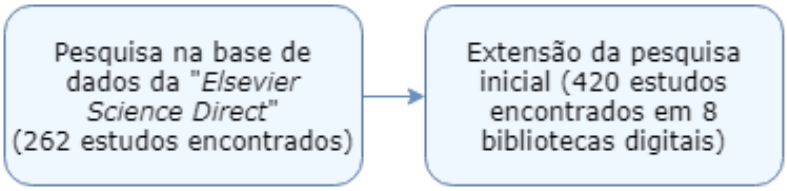

Figura 4.2: Etapas do Processo de Pesquisa

A definição de uma *query* de pesquisa. Foi definida uma *query* com base nas palavras-chave escolhidas, e realizada uma busca na base de dados digital selecionada no nosso estudo, por consenso, pelos especialistas da *cHiPSet ICT COST Action*, a *Elsevier Science Direct*. Pelo facto dos autores dos estudos primários nem sempre utilizarem as mesmas palavras-chave e por ser necessário cobrir o maior número de estudos relevantes possível, tentando não descartar potenciais documentos importantes, foram utilizados sinónimos das diferentes expressões nela contidas:

(“Big Data” OR “Data Intensive” OR “Stream Data”) AND (“Programming Model” OR “Language Model” OR “Modelling Language”) AND (“Domain Specific Language” OR “General Purpose Language” OR “Programming Language” OR “Programming Framework”) AND (“HPC” OR “High performance computing” OR “Grid Computing” OR “Supercomputing” OR “Parallel” OR “Concurrent”)

Com esta pesquisa da literatura, obtivemos **262 estudos**, candidatos a estudos primários a incluir no nosso EMS.

A extensão da pesquisa inicial. As referências encontradas foram, então, apresentadas ao conjunto de profissionais de domínio da *cHiPSet ICT COST Action*, para que aferissem a sua completude. Dessa análise, resultou que a cobertura da literatura oferecida na base de dados da *Elsevier Science Direct* era insuficiente, para este domínio, havendo um conjunto significativo de publicações relevantes que não fazia parte desta seleção,

por não serem editadas por esta biblioteca digital. Como estratégia de mitigação desta limitação, efetuei uma segunda pesquisa, tanto na biblioteca digital já verificada como em outras bases de dados existentes, utilizando *queries* semelhantes à apresentada. Este procedimento resultou em milhares de artigos, tornando a sua análise uma tarefa impossível de realizar, com os recursos disponíveis. Para resolver este problema, criei um separador na folha de cálculo partilhada, para que os peritos da *cHiPSet ICT COST Action* pudessem identificar as conferências e revistas científicas que seriam relevantes para este estudo secundário. Todos os 19 revisores da equipa participaram na criação desta *shortlist* (Tabela 4.2). Quando o preenchimento da *shortlist* atingiu um ponto de saturação em que os profissionais que a visitavam indicavam que as conferências e revistas científicas mais relevantes já estavam a ser cobertas, parei este processo. Os resultados da segunda pesquisa foram, depois, filtrados na referida *shortlist*.

Tabela 4.2: *Shortlist* das conferências e revistas científicas selecionadas

Conferências	**Revistas Científicas**
GTC / GPGPU conference	*ACM Transactions on Parallel Computing*
IEEE International Parallel and Distributed Processing Symposium	*Concurrency and Computation Practice and Experience*
International Conference on Parallel Processing	*Future Generation Computer Systems*
International Conference on Supercomputing	*IEEE Computing in Science and Engineering*
International European Conference on Parallel and Distributed Computing	*Journal of Parallel and Distributed Computing*
International Supercomputing Conference	*Journal of Supercomputing*
Parallel Computing Conference	*Parallel Computing*
Principles and Practice of Parallel Programming	*Scientific Programming*
SIAM Conference on Parallel Processing for Scientific Computing	
Supercomputing Conference	

Tabela 4.3: Resultados obtidos com a segunda pesquisa

Biblioteca Digital	**Número de Publicações Obtidas**
academia.edu	1
ACM Digital Library	16
Compendex	6
Elsevier Science Direct	27
Google Scholar	32
IEEE Xplore	3
Research Gate	3
Springer Link	70
Total de artigos encontrados:	**158**

Na Tabela 4.3, estão definidas as bibliotecas digitais e o respetivo número de estudos obtidos, já com a remoção de documentos duplicados efetuada. À excepção da base de dados da biblioteca *Google Scholar*, onde apenas foi possível a exploração das diferentes combinações das palavras-chave referidas, foram realizadas consultas complexas nas bases de dados das oito bibliotecas digitais.

É importante mencionar que todas as pesquisas foram baseadas no **título, resumo e palavras-chave dos artigos publicados** e foram limitadas **entre os meses de publicação Janeiro de 2006 e Março de 2018**. Segundo a opinião dos especialistas, os estudos editados antes de 2006 poderão ser excluídos sendo, em princípio, cobertos todos os estudos necessários para este EMS, por crerem que esta foi a época onde começaram a ser publicados mais documentos sobre os tópicos da Computação de Alto Desempenho e *Big Data*.

Posto isto, após removidos os artigos duplicados, possuímos um total de **420 artigos (262 com a primeira pesquisa + 158 com a segunda)** que foram processados ao longo das fases seguintes deste EMS.

4.1.3 Seleção de Estudos

Nesta etapa, o grupo de revisores do EMS inspecionaram os estudos primários. Para tal, recorreu-se a um conjunto de critérios de inclusão e exclusão (Tabela 4.4), definido antes da pesquisa efetuada. Os revisores fizeram uma seleção manual, analisando **o título, o resumo e as palavras-chave** de cada um dos 420 estudos.

Após terminar a seleção manual das publicações, prosseguiram **152 artigos** para as fases futuras (referidos no Apêndice E), tendo sido os restantes 268 documentos considerados irrelevantes para este EMS.

Tabela 4.4: Critérios de Inclusão e Exclusão de artigos

Critérios de Inclusão	Critérios de Exclusão
O estudo aborda o tópico da CAD	A publicação não aborda o tema da CAD, tornando-se irrelevante
O estudo foi revisto por pares (*peer-reviewed*[2]) e publicado em revistas científicas, conferências ou *workshops*	O estudo não foi revisto por pares (resumos, tutoriais, editoriais, *slides*, palestras, demonstrações de ferramentas, cartazes, painéis, notas-chave, relatórios técnicos) O estudo foi revisto por pares mas não publicado em revistas científicas, conferências ou *workshops* (p.e. tese de doutoramento, livros, patentes)
O estudo foi escrito em Inglês	O estudo não foi escrito em Inglês
O estudo encontra-se eletronicamente acessível	O estudo não se encontra eletronicamente acessível
O estudo está relacionado com a área da Ciência da Computação	O artigo foi publicado antes do ano de 2006

4.1.4 Validação do Processo de Revisão

A avaliação da qualidade do processo de revisão é uma tarefa relevante. Para validar este processo, respondeu-se a quatro questões fundamentais, referidas em [KC07; Kit+09]:

- **Os critérios de inclusão e exclusão dos estudos estão descritos e são apropriados?** Sim, este aspeto foi endereçado durante a etapa de *Seleção dos Estudos* (Subsecção 4.1.3). Todos os critérios considerados apropriados, por uma equipa de 19 membros da *cHiPSet ICT COST Action*, foram explicitamente definidos (Tabela 4.4), filtrando os iniciais 420 estudos num conjunto de 152 artigos;

- **A pesquisa literária cobriu todos os estudos relevantes?** Sim, este aspeto foi endereçado durante o *Processo de Pesquisa* (Subsecção 4.1.2). Inicialmente, existia uma lista preliminar de estudos candidatos que foi utilizada para validar a cobertura da pesquisa. Essa lista foi avaliada por profissionais de domínio, que concluiram que esta estava bastante incompleta, face a artigos que estes detetaram como ausentes. Como estratégia de mitigação desta limitação, efetuei uma segunda pesquisa nas plataformas electrónicas: *academia.edu*, *ACM Digital Library*, *Compendex*, *Elsevier Science Direct*, *Google Scholar*, *IEEE Xplore*, *Research Gate* e *Springer Link*, que resultou em milhares de artigos. Pelo facto da sua análise ser uma tarefa impossível de

realizar, com os recursos disponíveis, pedi a contribuição dos colegas da *cHiPSet ICT COST Action* para a criação de uma *shortlist* (Tabela 4.2) das conferências e revistas científicas consideradas relevantes para o estudo, que apoiassem as respostas às questões propostas. Quando terminaram essa lista, filtrei os resultados da minha pesquisa nessas conferências e revistas científicas;

- **Os revisores avaliaram a qualidade do estudo?** Sim, cada um dos estudos foi inspecionado por, pelo menos, um dos autores deste EMS. Apesar de não existir uma garantia absoluta de que não se estão a perder informações importantes, como mecanismo de salvaguarda, numa amostra de cerca de 15 artigos, um dos especialistas fez uma segunda revisão desses documentos, avaliando se estariam a ser excluídos dados importantes no nosso estudo, tendo-se concluído que a extracção estaria a ser feita de forma correcta;

- **Os estudos primários foram adequadamente referidos?** Ao longo do EMS, todos os estudos publicados pelas conferências e revistas científicas indicadas com a contribuição dos especialistas, que se enquadrassem no nosso projecto, foram identificados. Para além desses estudos, foi-lhes pedido que criassem uma *shortlist* de publicações (incluídas no conjunto de 420 estudos, referidos na Subsecção 4.1.2) que deveriam estar a ser encontradas e que poderiam não estar a ser.

4.1.5 Processo de Extração de Informação

Conforme o referido na Secção 3.2, de modo a preservar a consistência da extração dos dados, foi criado e implementado um formulário de recolha de informação. Neste momento do processo de revisão, aquando da análise de cada estudo, foi preenchido este formulário. Todos os dados recolhidos foram armazenados num separador, criado para o efeito, duma folha de cálculo partilhada. Os dados iniciais são referentes ao revisor da publicação e à própria publicação. A restante informação diz respeito às questões de investigação referidas na Subsecção 4.1.1. Nesta etapa, foi efetuada a análise pormenorizada dos estudos incluídos nas etapas antecedentes a esta, sendo extraída a informação considerada relevante, pelos respetivos revisores de cada um dos estudos (revisores atribuídos de modo aleatório), e dadas respostas às questões de investigação. Para além dos documentos atribuídos a cada um dos autores deste EMS, uma amostra de cerca de 15 estudos foi analisada por mais do que uma pessoa, como mecanismo de salvaguarda, de modo a avaliar se estariam a ser excluídos dados importantes no nosso estudo, tendo-se concluído, como já referido, que a extracção estaria a ser feita de forma correcta. Participaram neste processo os 19 membros do grupo de trabalho, sendo estes autores do EMS.

O Apêndice C trata-se do exemplo de um formulário que preenchi referente a um dos estudos incluídos ([Mis+18]).

4.1.6 Sumarização da Informação recolhida

Durante esta etapa, fiz a compilação da informação, anteriormente, extraída, identificando quais os estudos que eram relevantes, tendo em conta as questões de investigação formuladas. As ferramentas encontradas são integradas em LPG, que exigem experiência em programação, por parte de quem as utiliza. Pelo facto destas não facilitarem a tarefa de codificação para os especialistas de domínio, foram excluídos os estudos referentes às mesmas, tendo sido incluídos apenas aqueles que referiam, especificamente, linguagens utilizadas para a CAD.

Criei um documento partilhado onde respondi às questões gerais propostas e referi as linguagens encontradas, assim como os dados individuais de cada uma delas. Quando terminei o documento, os restantes elementos da *cHiPSet ICT COST Action* confirmaram o seu conteúdo, com a finalidade de verificar que as questões tinham sido corretamente respondidas e que nenhum estudo tinha sido, erradamente, excluído.

No Apêndice D, encontra-se sumarizada a informação referente às linguagens identificadas.

4.1.7 Estratégia de comunicação

Similarmente à fase de sumarização da informação, escrevi o documento, pedindo aos restantes elementos da rede, após efetuar essa tarefa, que o completassem e confirmassem todos os detalhes nele contidos.

Após escrito o EMS, é essencial a comunicação eficaz dos resultados deste estudo, para que se consiga, realmente, cumprir o objetivo de apoiar estudantes, investigadores, ou outros profissionais que necessitem de uma introdução ou uma visão panorâmica sobre as linguagens disponíveis para a CAD, no momento de decisão sobre quais são as linguagens mais promissoras a explorar, num determinado contexto. Por esse motivo, é preciso recorrer a formas de divulgação específicas de documentos desta área. Com a finalidade de se chegar a um consenso sobre qual a melhor conferência ou revista científica a contactar, para submeter o EMS, pedi a todos os colegas que criassem uma lista daquelas que considerassem relevantes. Para além das identificadas na Tabela 4.2, onde foram filtrados os resultados da segunda pesquisa, esta lista incluía: *Computing*; *ACM Computing Surveys*; *IEEE Transactions on Parallel and Distributed Systems*; *International Conference for High Performance Computing, Networking, Storage and Analysis*; *International Journal of High Performance Computing Applications*; *IEEE Micro* e *Journal of Computational Science*. Criei uma votação com todas as conferências e revistas científicas indicadas nessa lista, tendo sido escolhida a *Parallel Computing* como o melhor meio de divulgação. Na tomada das suas decisões, os votantes tiveram em conta factores como as suas experiências pessoais, no papel de membro do quadro editorial, revisor ou autor, com cada uma das conferências e revistas científicas. Foi contactada a revista científica escolhida, tendo esta aceite receber, com grande entusiasmo, o nosso EMS.

4.2 Discussão dos Resultados obtidos

Ao longo desta Secção, vão ser discutidas as respostas às Questões de Investigação (QI) propostas na Subsecção 4.1.1. Da informação sumarizada, das linguagens no geral, conseguimos extrair conclusões. As linguagens de programação utilizadas para a CAD, no processamento de *Big Data*, são inicialmente categorizadas na QI 1 e as suas caraterísticas analisadas ao longo das restantes questões.

Para além das informações recolhidas sobre as linguagens existentes, foram encontrados vários documentos referentes a bibliotecas (p.e. [Ald+17; EK10; Sjö+16; Viñ+17]), integradas nas linguagens referidas, *Application Programming Interfaces (API)* (p.e. [BC17; Sen+15; Wan+14]) e modelos de programação (p.e. [Ded+14; Jin+17; Sim+15; Van02]). No final deste processo, identifiquei **33 artigos**, que referem **26 linguagens**.

O Apêndice D apresenta a lista das linguagens identificadas, descrevendo as suas caraterísticas. Devido à similaridade das respostas dadas às questões de investigação, algumas linguagens foram agrupadas, como: *C* e *C++*, ou *Python* e *R*.

4.2.1 Questão de Investigação 1 - Quais são as categorias das linguagens em utilização para a CAD?

As linguagens de programação encontradas foram separadas em quatro categorias:

1. Linguagem de Domínio Específico (LDE), que se trata de uma linguagem adaptada a um domínio de aplicação específico que oferece anotações e abstrações apropriadas [Kos+10; Kos+16; VD+00];

2. Linguagem de Propósito Geral (LPG), sendo uma linguagem de programação projetada para ser utilizada na escrita de *Software* numa ampla variedade de domínios de aplicação [Kos+10];

3. LDE embebida numa outra LDE;

4. LDE embebida numa LPG.

Tabela 4.5: Total de linguagens encontradas de cada categoria

Categoria da Linguagem	Número de Linguagens Encontradas
LDE	8
LPG	14
LDE embebida numa outra LDE	0
LDE embebida numa LPG	4
Total de linguagens descobertas:	**26**

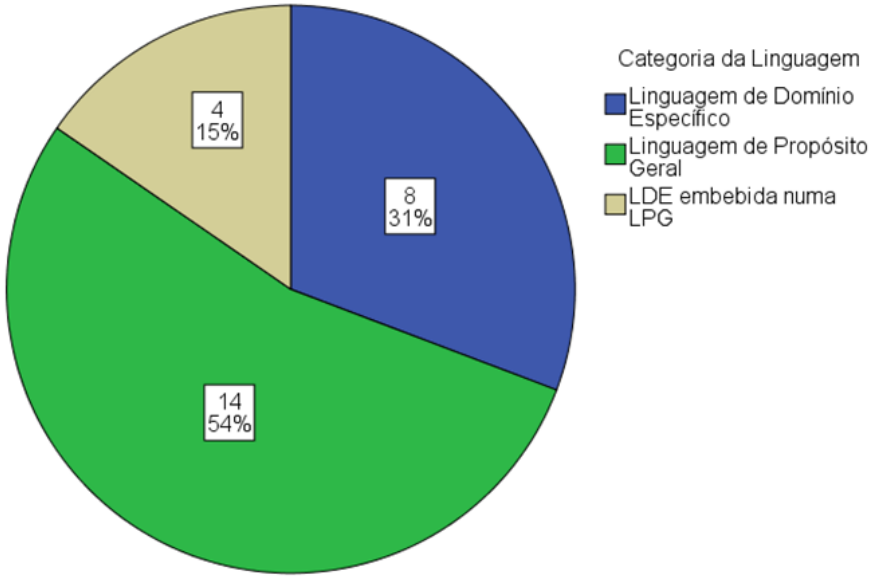

Figura 4.3: Quais são as categorias das linguagens em utilização para a CAD? - QI 1

Segundo os resultados apresentados na Tabela 4.5 e na Figura 4.3, onde é mostrado o total e percentagem de linguagens encontradas de cada categoria, conclui-se que **54%** das linguagens focadas nas publicações encontradas foram classificadas como LPG (**14 linguagens**) e **31%** (**8 linguagens**) como LDE. As restantes (**4 linguagens**) foram consideradas LDE embebidas numa LPG.

A nossa análise não encontrou nenhum estudo que referisse uma LDE embebida numa outra LDE.

4.2.2 Questão de Investigação 2 - Qual é a natureza das linguagens para a CAD?

Na Secção D.1, encontra-se sumarizada, referente a cada linguagem, a informação encontrada que dá resposta à presente questão de investigação.

As linguagens encontradas utilizam *Virtual Execution Environments*, como *Java Virtual Machine (JVM)*, e *Distributed Middlewares*, como *Hadoop Distributed File System (HDFS)* e *IBM InfoSphere*.

A principal vantagem das linguagens é a sua usabilidade. A facilidade de configuração, a portabilidade, a orquestração e o desempenho são as outras vantagens que foram encaradas como importantes (Figura 4.4).

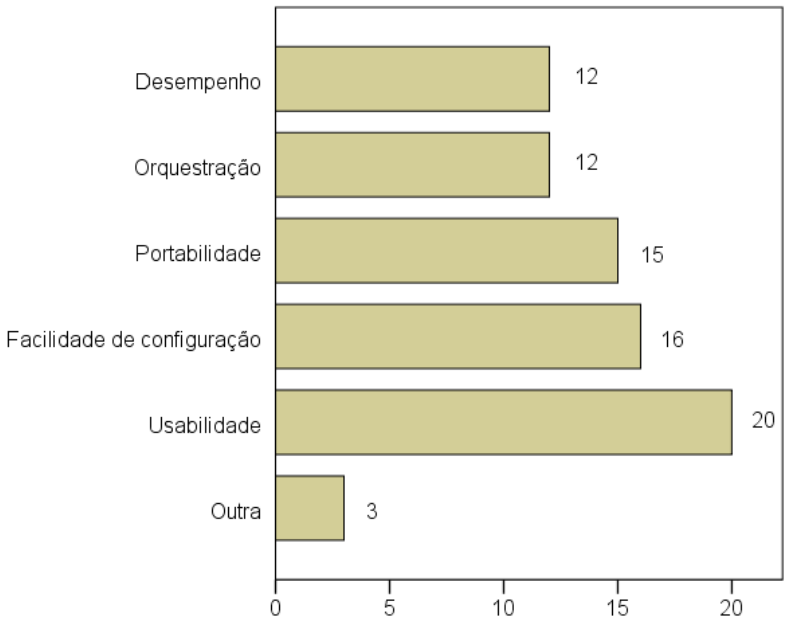

Figura 4.4: Quais são as principais vantagens da linguagem? - QI 2.3

A maioria é orientada a objetos. Este paradigma é seguido pelo funcional (Figura 4.5).

Geralmente, estas suportam múltiplos sistemas operacionais. Algumas requerem *Message Passing Middleware*, como *Message Passing Interface (MPI)*; arquiteturas *IO*, como *HDFS*; e bibliotecas, como *Apache Hadoop*, para suportar os artefatos criados.

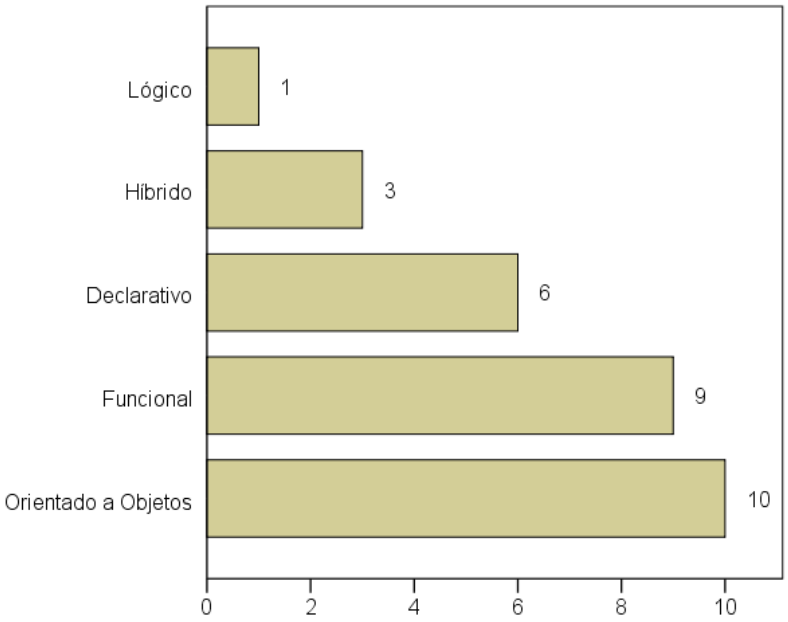

Figura 4.5: Quais são os paradigmas subjacentes às linguagens? - QI 2.5

Relativamente ao suporte de ferramentas existente para as linguagens, os compiladores são a ferramenta melhor representada, seguida de *tool suite* e dos interpretadores (Figura 4.6).

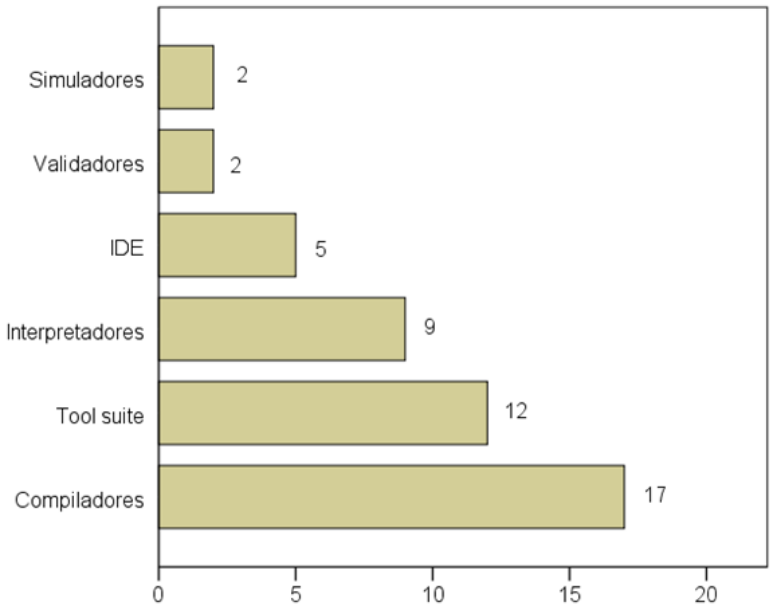

Figura 4.6: Qual é o suporte de ferramentas existente para a linguagem? - QI 2.7

Existem algumas tecnologias utilizadas para criar o *tool suite* das linguagens. Mais especificamente, *compiler generators*, como *IBM Infosphere Streams*; algumas tecnologias baseadas em *XML* (p.e. em *Java*, recorreu-se a uma sintaxe para descrever classes e métodos); e alguns *frameworks*, como *Knowledge Discovery Toolbox (KDT)*.

Grande parte das linguagens encontradas não suporta *hardware* específico (**85%**). Sabe-se que **40%** das linguagens suporta *GPUs* ou arquiteturas *multi-core*.

O principal propósito das linguagens é a implementação da solução, seguido da formalização da solução, da formalização dos requisitos do problema e da interpretação dos dados (Figura 4.7).

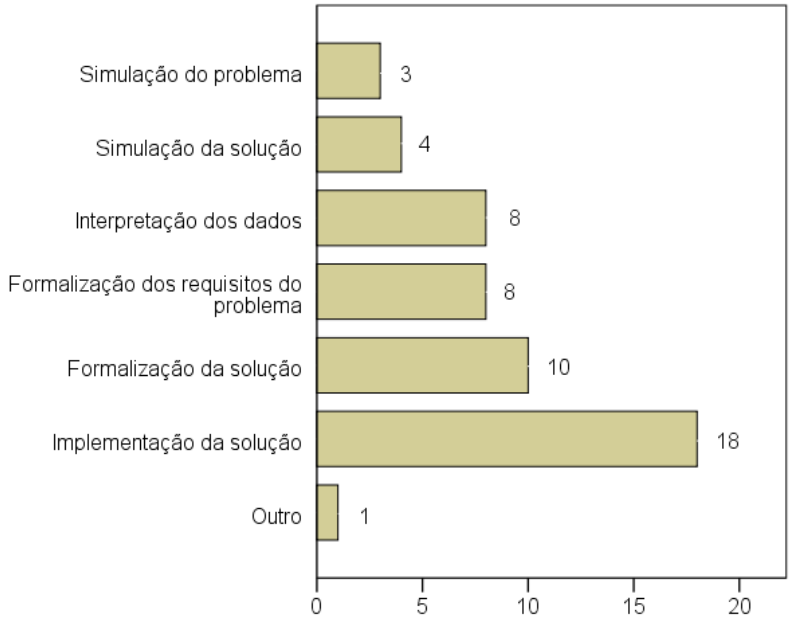

Figura 4.7: Qual é o propósito da linguagem? - QI 2.10

Os resultados para o tipo de representação da linguagem revelaram que existe uma sintaxe concreta para todas as linguagens, sendo o tipo de representação preferencial de **76%** delas o textual (Figura 4.8).

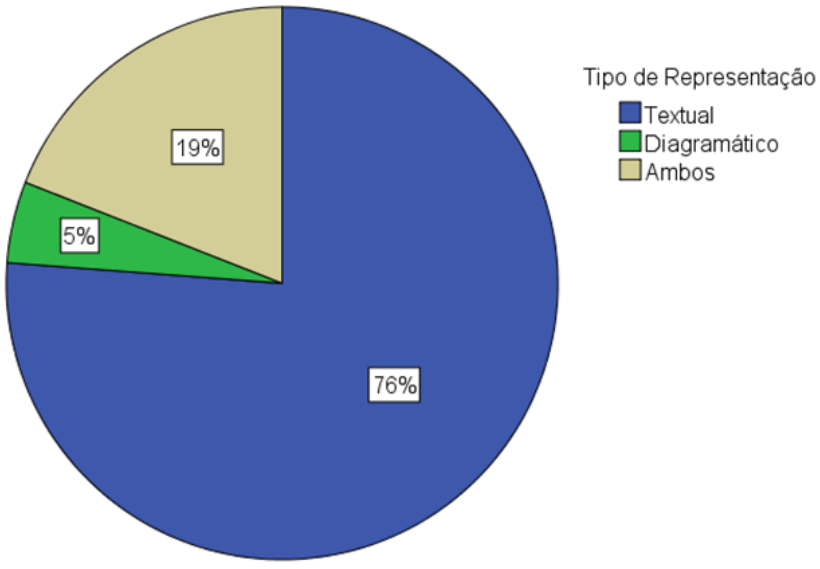

Figura 4.8: Qual é o tipo de representação preferencial da linguagem? - QI 2.11

4.2.3 Questão de Investigação 3 - Quais são os perfis de utilizador típicos para as linguagens?

Na Secção D.2, encontra-se sumarizada, referente a cada linguagem, a informação encontrada que dá resposta à atual questão de investigação.

A Figura 4.9 mostra a distribuição dos perfis de utilizador típicos para as linguagens identificadas. São os utilizadores finais que mais recorrem a estas linguagens, utilizando-as para resolver problemas. Sabe-se que **16,5%** das linguagens são empregadas por desenvolvedores, para a criação de ferramentas para terceiros.

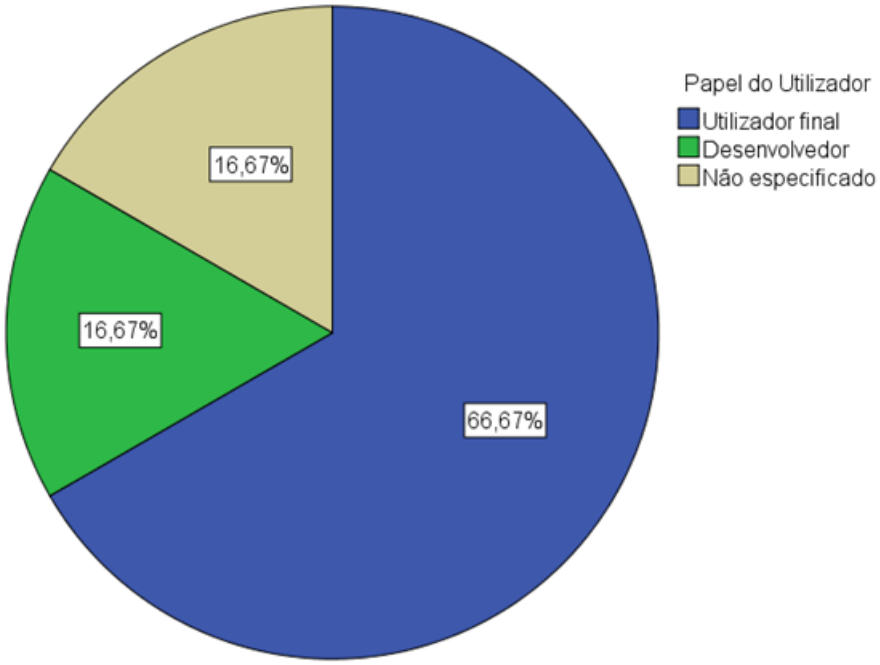

Figura 4.9: Quais são os papéis dos utilizadores desta linguagem? - QI 3.1

O conhecimento técnico necessário para utilizar estas linguagens é baseado, principalmente, nas próprias linguagens ou naquelas onde estão embebidas, alguns conhecimentos de *hardware*, *clusters* e informação teórica relacionada com o domínio da aplicação.

4.2.4 Questão de Investigação 4 - Quão eficazes são as linguagens?

Na Secção D.3, encontra-se sumarizada, referente a cada linguagem, a informação encontrada que dá resposta a esta questão de investigação.

A eficácia é articulada em três aspectos, abordados pela QI 4.1 (sucesso), QI 4.2 (ganho de produtividade) e QI 4.3 (vantagem contra as abordagens concorrentes).

Conforme mostrado na Figura 4.10, o sucesso das linguagens é avaliado na maioria dos artigos analisados.

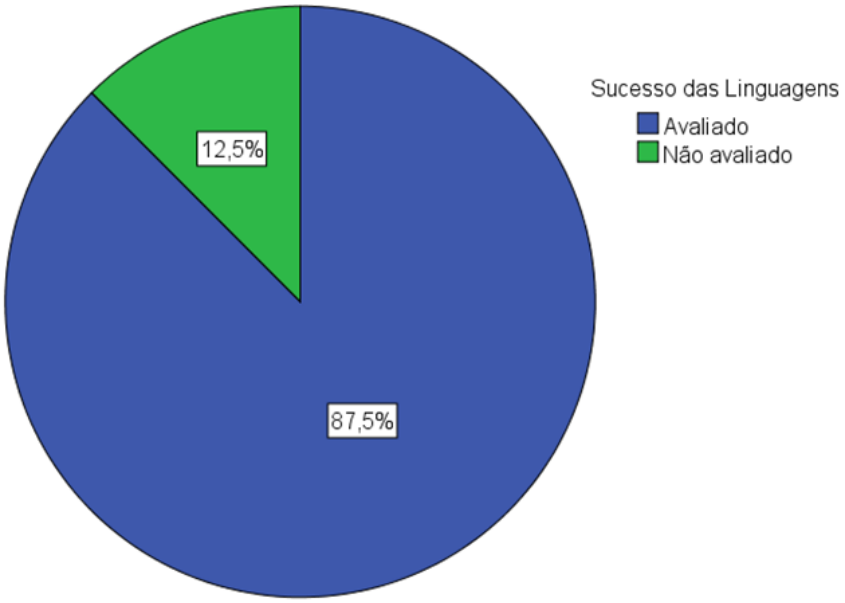

Figura 4.10: O sucesso das linguagens é avaliado no artigo? - QI 4.1

A Figura 4.11 mostra uma representação gráfica dos ganhos de produtividade trazidos pelas linguagens reportadas. Com uma **análise quantitativa**, o ganho de produtividade mais referido foi a facilidade de utilização da linguagem, seguido pela facilidade de aprendizagem da mesma. Com uma **análise qualitativa**, o ganho de produtividade mais referido foi a facilidade de utilização, seguido pela baixa sobrecarga cognitiva e pela facilidade de aprendizagem da linguagem. Comparando os dois tipos de análise anteriores, os ganhos de produtividade trazidos pelas linguagens relatadas foram, principalmente, avaliados recorrendo a **métodos qualitativos**, o que poderá revelar a oportunidade de escrita de mais estudos que meçam estes ganhos de modo quantitativo.

O gráfico da Figura 4.12 apresenta os ganhos de desempenho trazidos pelas linguagens reportadas. Com uma **análise quantitativa**, o ganho de desempenho mais referido foi a eficiência de computação e a escalabilidade da solução. Com uma **análise qualitativa**, o ganho de desempenho mais referido foi a capacidade de evolução/manutenção, seguido pela escalabilidade da solução. Contrariamente ao ocorrido relativamente aos ganhos de produtividade, em comparação com os dois tipos de análise anteriores, os ganhos

de desempenho trazidos pelas linguagens reportadas foram, principalmente, avaliados recorrendo a **métodos quantitativos**.

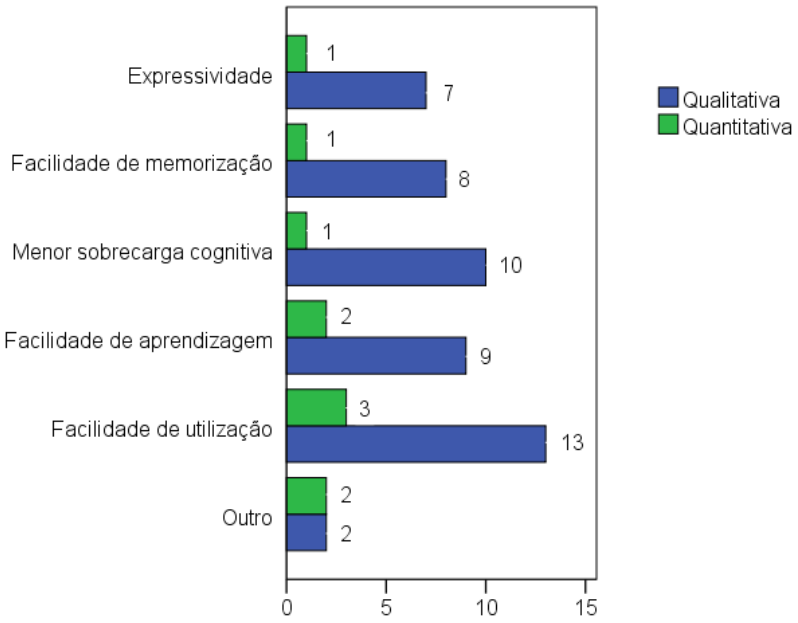

Figura 4.11: Quais são os ganhos de produtividade trazidos pelas linguagens reportadas e que tipo de medição foi utilizado? - QI 4.2

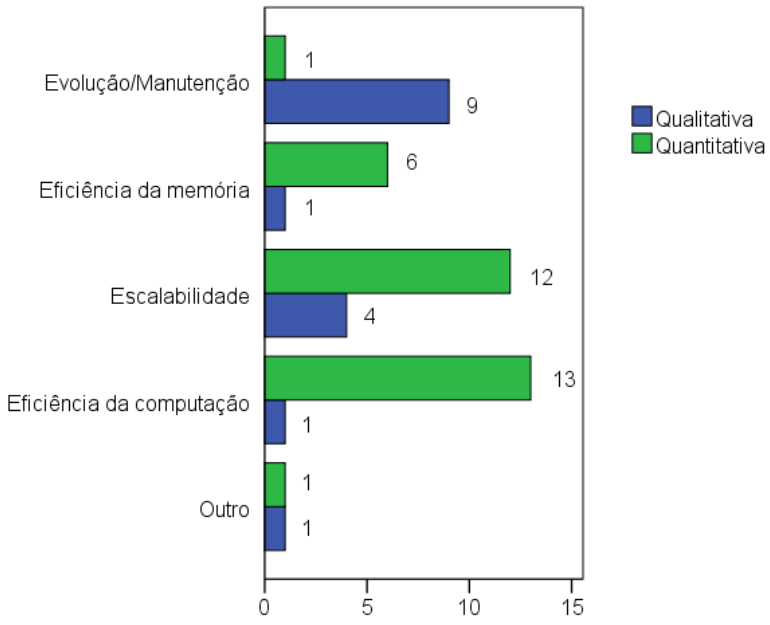

Figura 4.12: Quais são os ganhos de desempenho trazidos pelas linguagens reportadas e que tipo de medição foi utilizado? - QI 4.2

De acordo com as estatísticas, **64%** dos artigos incluíram uma comparação explícita entre a linguagem encontrada e abordagens concorrentes. Metade das publicações incluiu uma comparação explícita da linguagem proposta com respeito a contextos/configurações distintas.

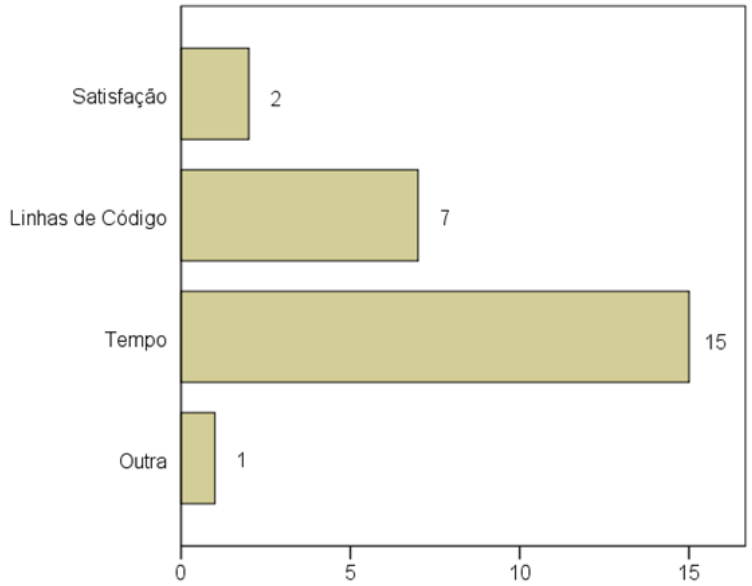

Figura 4.13: Número de artigos a utilizar cada métrica - QI 4.3: Quais são as métricas utilizadas?

Todas as métricas foram analisadas utilizando métodos quantitativos.

A métrica mais utilizada foi o tempo computacional, seguido pelas linhas de código e pela satisfação de utilização da linguagem (Figura 4.13).

4.2.5 Questão de Investigação 5 - Que tipos de artigos são publicados na área de modelos de programação para a CAD?

Uma grande parte das publicações que referiam linguagens para a CAD não incluem autores da *cHiPSet ICT COST Action*, mas existem algumas exceções como [Ald+17; Men+17; Mis+18].

A revista científica que publicou mais artigos foi "*Future Generation Computer Systems*", seguida por "*Parallel Computing*" e por "*Journal of Parallel and Distributed Computing*", como ilustrado na (Figura 4.14).

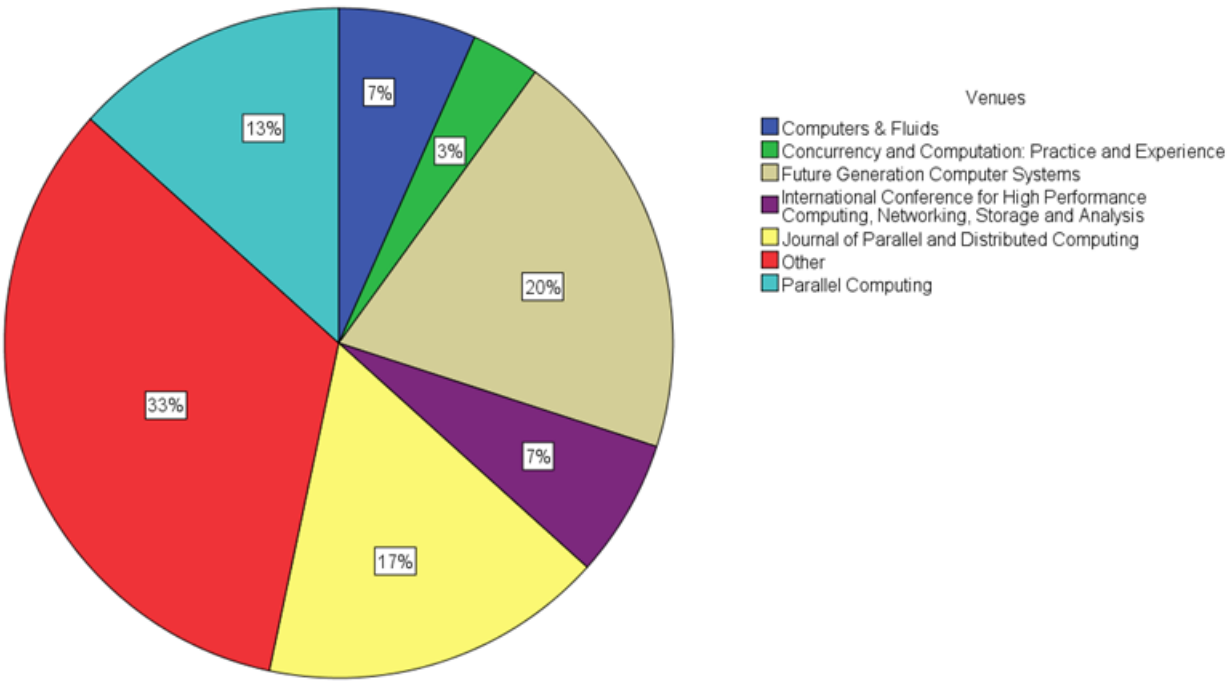

Figura 4.14: Que conferências e revistas científicas publicaram artigos sobre linguagens para a CAD? - QI 5.3

A maioria desses artigos foi patrocinada por fundos públicos ou públicos e privados. Dado que cada um destes pode ser classificado tanto como estudo de caso, como relatório de experiência e/ou avaliação comparativa, verificou-se uma maior ocorrência de relatórios de experiências. Também foram encontrados vários estudos de caso e avaliações comparativas em número semelhante, como mostrado na Figura 4.15.

De acordo com os valores apresentados na Tabela 4.5 e na Figura 4.3, 54% das linguagens encontradas são do tipo LPG. Foram descobertos vários artigos referentes a LDE e três publicações que diziam respeito a LDE embebidas em LPG. Na Tabela 4.6, estão enumeradas estas linguagens.

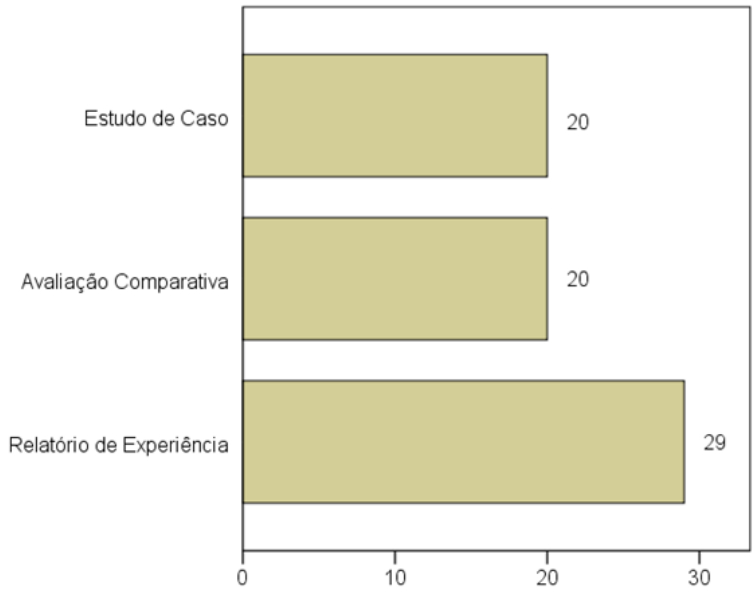

Figura 4.15: Número de artigos que reportam cada tipo de pesquisa - QI 5.5

Tabela 4.6: Linguagens encontradas com a pesquisa efetuada

Categoria da Linguagem	Nome da Linguagem
Linguagens de Domínio Específico	*CineGrid Description Language* [Kon+11]; *Crucible* [Coe+14]; *e-Science Central WFMS* [Cal+16]; *Higher-order "chemical programming" language* [Fer+14]; *Liszt* [DeV+11]; *Mendeleev* [CJ17]; *MiniZinc* [Cab+15]; *Network Description Language* [Kon+11]
Linguagens de Propósito Geral	*Bobolang* [Fal+14]; *C* [Bad+15; Bin+13; EK10; Lia+16; Mea+17; Obr+12; Sen+15]; *C++* [Bad+15; Bin+13; EK10; Lia+16; Mea+17; Obr+12; Sen+15]; *Erlang* [Tur+16]; *FastFlow* [Ald+17; Men+17]; *Goal Language (supported by RuGPlanner)* [Kal+16]; *Java* [Bad+15; Car+13; Mat+10; Mat+11; Mea+17]; *OpenCL* [Bin+13; Kim+15]; *Python* [Bad+15; Hin+06; Luc+15]; *R* [Bad+15; Hin+06; Luc+15]; *Scout* [McC+07]; *Selective Embedded Just-In-Time Specialization* [Lug+15]; *SkIE-CL* [CV02]; *Swift* [Mah+16; Wil+11]
LDE embebidas em LPG	*Pipeline Composition* [Mis+18]; *Spark SQL* [Liu+15]; *Spark Streaming* [Liu+15]; *Weaver* [Bui+11]

4.3 Avaliação do EMS por Profissionais

Preparei um questionário como forma de validação dos resultados encontrados e confrontamento com a opinião do que os profissionais inseridos na área da CAD esperavam encontrar. O questionário foi baseado nas questões de investigação propostas para este EMS e explicitadas na Subsecção 4.1.1.

Este questionário é apresentado no Apêndice A e visa descobrir: em que áreas da engenharia trabalham os especialistas; se a sua atividade consiste primeiramente no desenvolvimento de ferramentas de suporte ou no manuseamento de ferramentas já existentes; que linguagens de programação são empregues nesta área; o que os faz utilizar essas linguagens em relação às restantes que conhecem (no contexto em questão); quais são as vantagens dessas linguagens; quais são as ferramentas de suporte existentes que conhecem; para o domínio onde estão inseridos, quão eficazes consideram as linguagens utilizadas, isto é, qual é o seu grau de sucesso na produção de um resultado desejado; quais são os ganhos de desempenho trazidos pelas linguagens reportadas; quais são as principais limitações/dificuldades de utilização das linguagens identificadas.

4.3.1 Resultados do questionário

Ao longo da presente Secção, será apresentada a síntese dos resultados extraídos da análise das respostas dadas ao questionário.

Tendo em conta as respostas, é possível concluir que, com uma larga experiência na CAD dos inquiridos (afirmam trabalhar na área há mais de 10 anos e consideram ter um nível elevado de conhecimento técnico para as linguagens utilizadas (Figura 4.16)):

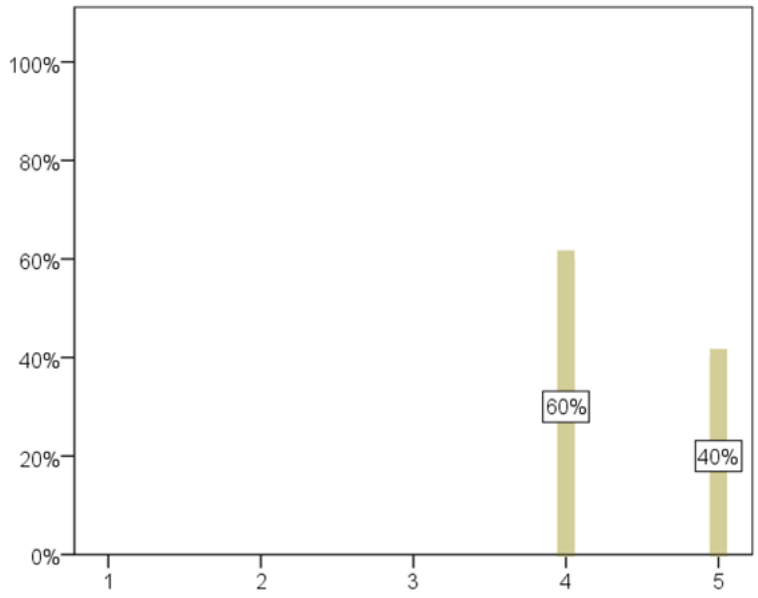

Figura 4.16: Como classifica o seu nível de conhecimento técnico para as linguagens utilizadas para a CAD? - Questão 14

- A sua atividade consiste, principalmente, no **desenvolvimento de novas ferramentas de suporte**, em vez de na utilização de ferramentas já existentes (Figura 4.17);

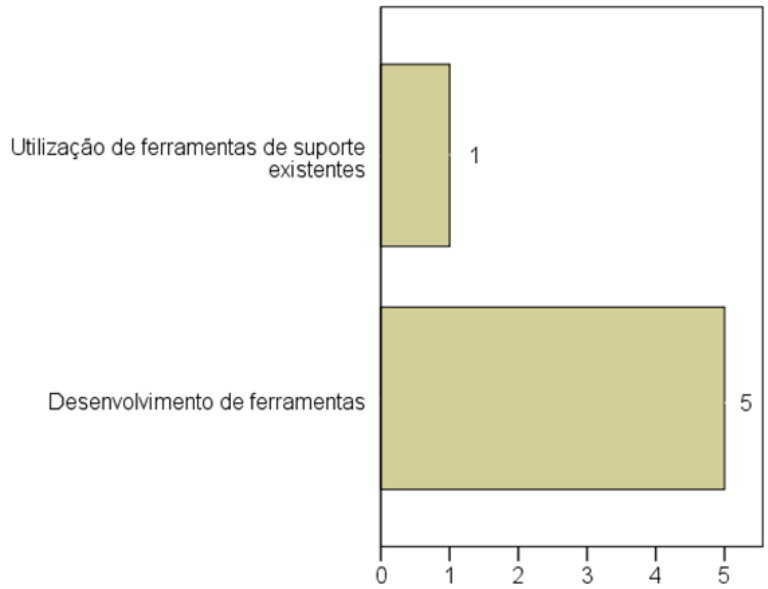

Figura 4.17: A sua atividade consiste primariamente no desenvolvimento de ferramentas de suporte ou na utilização de ferramentas já existentes? - Questão 4

- Todos utilizam as linguagens de programação ***C***, ***C++*** e ***OpenCL***, sendo que, também, são exploradas as seguintes: ***Java***, ***Python*** e ***R***;

- **A usabilidade e a natureza do problema em questão** são as principais razões que os fazem utilizar as linguagens referidas em relação às restantes que conhecem (Figura 4.18);

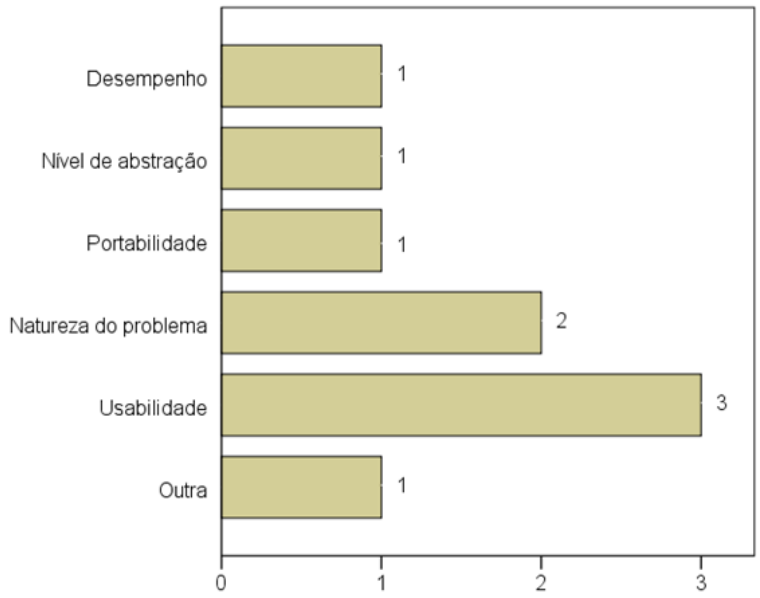

Figura 4.18: O que o faz utilizar essas linguagens em relação às restantes que conhece? - Questão 6

- **A portabilidade, o desempenho e a usabilidade** das linguagens referidas são as principais vantagens apontadas pelos especialistas (Figura 4.19);

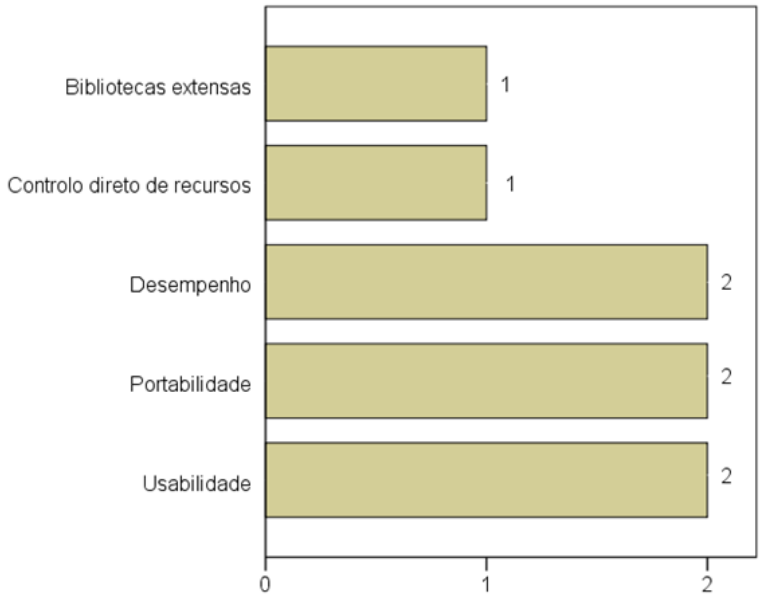

Figura 4.19: Quais são as principais vantagens dessas linguagens? - Questão 7

- O suporte de ferramentas existente para as linguagens utilizadas foram avaliadas **com uma moda de 3** (de 1 a 5, onde 1 indica que esse suporte é muito pobre e 5 que é excelente, Figura 4.20), e as ferramentas de suporte existentes mencionadas são: ***VAMPIR, CUDA SDK, Performance API (PAPI)*** **e** ***Linux performance tools*** (Figura 4.21);

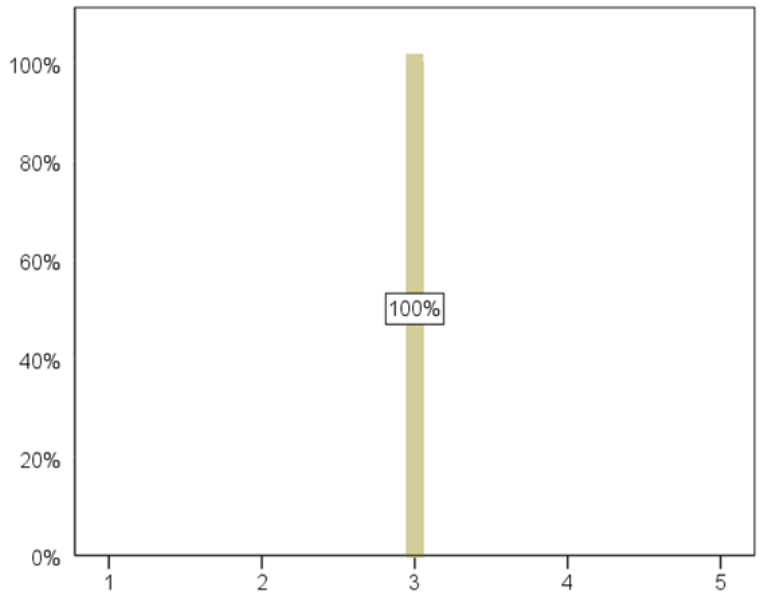

Figura 4.20: Como classifica o suporte de ferramentas existente para as linguagens que utiliza para a CAD? - Questão 8

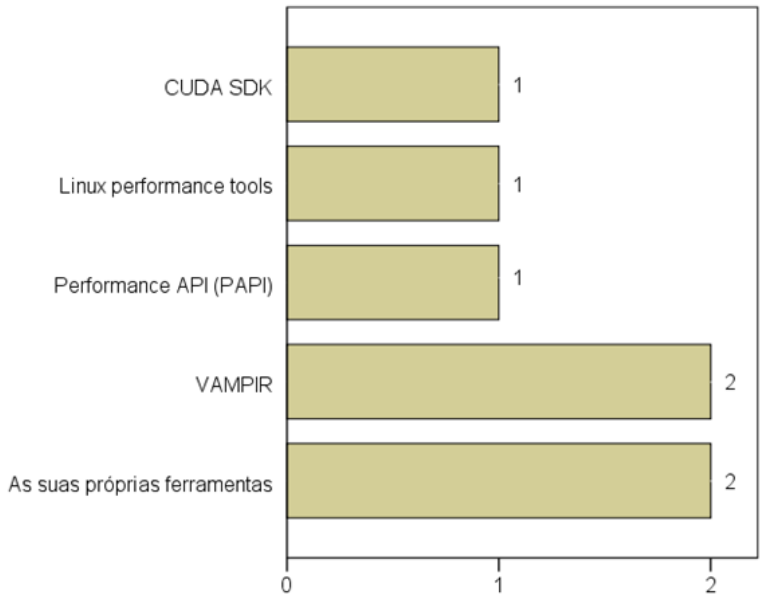

Figura 4.21: Em relação à questão anterior, qual é o suporte de ferramentas existente que conhece? - Questão 9

- As linguagens utilizadas foram consideradas **eficazes, com uma média de 3,6** (onde 1 indica que não são eficazes e 5 que são extremamente eficazes, Figura 4.22), e os mecanismos fundamentais da linguagem que justificam essa decisão são **o suporte para paralelismo de dados e o controlo direto de recursos, como a memória**;

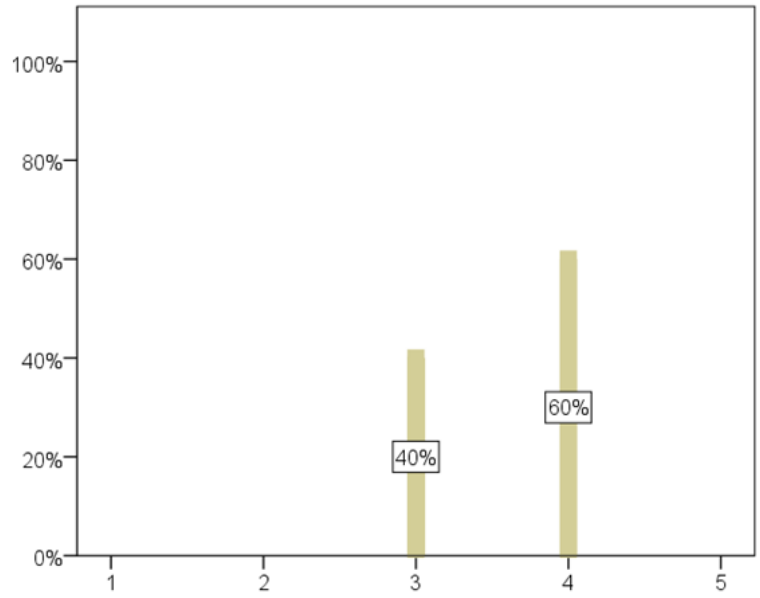

Figura 4.22: Para o domínio onde está inserido, como classifica a eficácia das linguagens que utiliza? - Questão 10

- Tendo em conta um leque de profissionais diverso, foram identificados vários obstáculos mas, de acordo com as respostas fornecidas, **a interoperabilidade e a curva de aprendizagem** são as principais dificuldades sentidas (Figura 4.23).

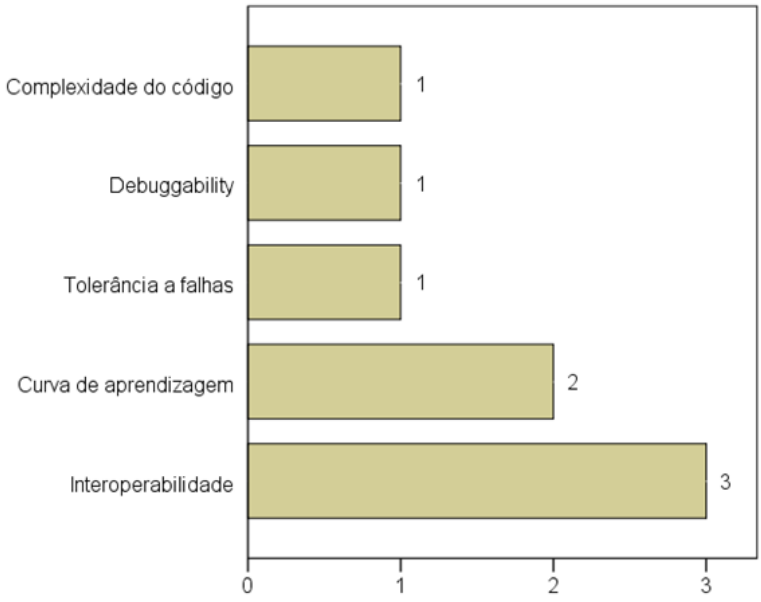

Figura 4.23: Quais são as limitações/dificuldades das linguagens que utiliza?-Questão 13

4.3.2 Comparação entre a informação extraída dos artigos e a análise de dados dos questionários

Comparando os resultados obtidos pela análise das respostas dadas ao questionário com as linguagens encontradas através da pesquisa efetuada, confrontando com a opinião do que era esperado encontrar, segundo os profissionais, é sabido que existem muitas linguagens utilizadas para a CAD, assinaladas na Secção 4.2, e, por esse motivo, uma parte delas não era conhecida pelos inquiridos. **As linguagens referenciadas na maioria dos artigos, as mais populares, são as linguagens utilizadas por estes especialistas: *C*, *C++*, *Java*, *OpenCL*, *Python* e *R***.

Os documentos referem um número de vantagens das linguagens mais vasto do que os peritos, incluindo a facilidade de configuração e a orquestração. No entanto, **as principais vantagens apontadas por estas pessoas, como a portabilidade, o desempenho e a usabilidade das linguagens, foram encontradas ao longo das publicações**. Uma vez que sabemos as vantagens das linguagens, de acordo com a informação encontrada nas bibliotecas digitais, **as razões, referidas pelos profissionais, que os fazem utilizar as linguagens referidas em relação às restantes que conhecem foram as previstas**.

Após a anterior comparação, é possível afirmar que, apesar da baixa quantidade de inquiridos, os resultados vêm em linha com o encontrado nos estudos.

4.4 Ameaças à Validade deste Estudo

Uma das mais destacáveis ameaças do nosso estudo e de qualquer outro estudo secundário é saber se foram encontrados todos os estudos relevantes para o seu desenvolvimento [KC07; Kit+10]. Para tentar minimizar esse problema, fizemos uma pesquisa inicial, apenas na base de dados de uma biblioteca digital, *Elsevier Science Direct*, chegando à conclusão de que a sua cobertura era insuficiente pois, parte da literatura estava a ser excluída, o que levou à condução de uma segunda pesquisa, em 8 bibliotecas digitais, atualizando a informação, anteriormente, recolhida. Para tal, foi pedido aos colegas da rede que criassem uma *shortlist* das conferências e revistas científicas relevantes para o nosso caso, de modo a podermos filtrar os resultos obtidos com as nossas consultas às bases de dados, pelo facto de se terem obtido milhares de estudos primários, tornando a sua análise uma tarefa impossível de realizar, com os recursos disponíveis. De modo a complementar a nossa investigação, para além dessa tarefa, foi pedido o contributo dos elementos da *cHiPSet ICT COST Action*, para a criação de uma *shortlist* de publicações que deveriam estar a ser detetadas pelas pesquisas e que poderiam não estar a ser. É sabido que uma pequena parte dos artigos listados não foi encontrada com a investigação feita, mesmo sendo os documentos publicados pelas conferências ou revistas científicas da lista referida anteriormente, o que leva a concluir que as pesquisas automáticas não conseguem ser totalmente confiáveis [Kit+10].

Recorrendo aos critérios de seleção dos estudos, foram excluídas aquelas publicações que: não tinham sido revistas por pares; tinham sido revistas mas não publicadas em conferências ou revistas científicas, como é o caso das teses; não estavam escritas em Inglês; não estavam eletronicamente acessíveis. Assumiu-se que a literatura de boa qualidade compreenderia estas condições, particularmente agora que o interesse por este tipo de estudos tem vindo a aumentar [KC07; Kit+10].

É de ter em conta que existe uma probabilidade de não encontrar alguns estudos, por não utilizarem as palavras-chave incluídas nas nossas *strings* de busca [Woh14b]. De modo a tentar combater este problema, foram utilizados vários sinónimos de cada uma das expressões contidas na *query*.

A análise e extração de informação de quem classifica cada artigo é, por si só, uma ameaça. É devido a esta razão que estes estudos são realizados, idealmente, com mais do que um revisor por estudo primário, de modo a tentar não excluir informação crucial [KC07; Kit+10]. Como mecanismo de salvaguarda, numa amostra de cerca de 15 artigos, foi feita a análise por um segundo revisor. Dessa análise resultou que os dados estavam a ser bem recolhidos, pelo menos na amostra analisada, contribuindo para uma maior confiança no processo de extracção de informação.

O processo de *snowballing* consiste em adicionar a um conjunto inicial de estudos encontrados, aqueles referenciados por estes que pareçam ser importantes, até não existirem

novos ficheiros a adicionar ao procedimento [Woh14a]. Neste EMS, não recorremos ao processo referido porque, para além da grande amostra de artigos encontrados que, por essa razão, foram filtrados nas conferências e revistas científicas consideradas relevantes, estariam a ser adicionados novos estudos, sendo que os últimos apresentariam as mesmas linguagens que os anteriores, não trazendo, em princípio, novos dados. Para além disso, o nosso documento contém apontadores para os estudos primários incluídos, sendo dada a possibilidade aos leitores de analisar, em maior detalhe, os dados sobre cada uma das linguagens.

A reduzida quantidade de respostas obtidas ao questionário poderá ser, também ela, uma ameaça. Contudo, verificou-se que estas vêm em linha com o encontrado nos estudos. Por considerar este um número reduzido, será lançada uma nova versão do questionário.

Adicionalmente, é sabido que são criadas, constantemente, novas linguagens de programação, o que leva a que um estudo secundário como este nunca consiga ser totalmente completo nem atualizado.

4.5 Sumário

O objetivo principal desta dissertação é o de criar um EMS que procura dar uma visão geral das linguagens existentes para a CAD, com apontadores para estudos mais focados, referindo as diferentes linguagens identificadas e os estudos primários a elas associadas, detetados através da nossa pesquisa, e retirando conclusões sobre as caraterísticas globais das mesmas. Participei, ativamente, em todas as etapas do processo de revisão, em colaboração com os restantes membros da *cHiPSet ICT COST Action*, e coordenei todo o estudo, tendo um papel executivo em todas as tarefas constituintes do mesmo.

Inicialmente, realizei uma pesquisa de modo a confirmar que não existiam, até ao momento, documentos literários sobre as linguagens utilizadas neste tipo de computação e efetuada uma pesquisa sobre as ferramentas que auxiliam este tipo de estudos. Foram definidas as questões de investigação a ser respondidas, o processo de pesquisa dos artigos e o de seleção dos estudos relevantes. Posteriormente, validou-se o processo de revisão, extraíram-se as informações relevantes de cada estudo incluído e sumarizei essa informação definindo, após isso, uma estratégia de comunicação do EMS.

Paralelamente a estes passos, criei um questionário que teve como principal função confrontar os dados recolhidos com as linguagens que os especialistas utilizam, logo, que esperavam que fossem endereçadas. Concluiu-se que a informação fornecida pelos inquiridos foi encontrada, assim como, outras linguagens não utilizadas por estes.

Capítulo 5

Conclusões

Nos capítulos anteriores, foram: descrita a necessidade de conhecer as linguagens existentes para a CAD e o problema da literatura existente se encontrar muito dispersa; desenvolvida uma solução para o resolver: um EMS que identifica as linguagens atuais utilizadas para a CAD e agrega tais informações.

Foram formuladas cinco questões de investigação, subdivididas em 22 sub-questões. Procedeu-se a uma pesquisa realizada em duas etapas: a primeira foi efetuada na base de dados da biblioteca digital "*Elsevier Science Direct*"; a segunda utilizou oito bases de dados digitais distintas (incluindo a referida), mas os seus resultados foram filtrados na *shortlist* das conferências e revistas científicas consideradas relevantes para o tópico em questão. Apenas foram considerados os artigos publicados entre Janeiro de 2006 a Março de 2018. De um total de 420 artigos encontrados na pesquisa, apenas 152 foram considerados relevantes para o nosso estudo. A sua análise permitiu identificar 26 linguagens utilizadas para este tipo de computação, referidas em 33 publicações. Concluiu-se que a maioria das linguagens utilizadas na CAD, no processamento de *Big Data*, são classificadas como Linguagens de Propósito Geral (LPG) em vez de Linguagens de Domínio Específico (LDE) e são os utilizadores finais que mais recorrem às mesmas. Esta observação leva-nos a concluir que existe uma oportunidade de investigação aplicada de linguagens de programação que tornem a tarefa de codificação mais fácil para os especialistas do domínio.

Como mecanismo de validação e avaliação dos dados recolhidos, foi criado um questionário, cujo principal objetivo foi o de confrontar a informação recolhida dos artigos com a opinião do que os profissionais esperavam encontrar. Dessa comparação resultou que o conhecimento encontrado era o esperado, pelo facto das linguagens referidas pelos inquiridos terem sido as mais mencionadas nos estudos, assim como as suas principais vantagens apontadas.

Pelos resultados obtidos neste estudo secundário, é possível concluir que, apesar deste tópico já estar a ser coberto pela comunidade científica, terá de ser continuado, principalmente pelo facto de que a maioria dos estudos primários encontrados apenas se foca nas bibliotecas (p.e. [Ald+17; EK10; Sjö+16; Viñ+17]) integradas nas linguagens referidas, que não deveriam ser consideradas mais relevantes do que as próprias linguagens em que se baseiam, nas *Application Programming Interfaces (API)* (p.e. [BC17; Sen+15; Wan+14]) e nos modelos de programação existentes (p.e. [Ded+14; Jin+17; Sim+15; Van02]).

5.1 Contribuição

O grande contributo externo em que nos baseámos para a criação do EMS foram os artigos inseridos nas bases de dados das oito bibliotecas digitais onde foi efetuada a nossa investigação.

Para que se consiga um melhor desempenho dos programas, utilizando os múltiplos *cores*, é necessário tentar reescrever (pelo menos, parcialmente) os programas sequenciais, com a finalidade de se tornarem mais paralelos, vendo-se os desenvolvedores de *Software* obrigados a aprender a fazê-lo e, por suposto, a conhecer as linguagens existentes para facilitar a computação. Como tal, tendo em conta o facto de se tratar de uma área presente nos dias de hoje, a criação de um documento que agregasse as informações sobre essas linguagens seria muito vantajosa.

Na Secção 1.4, foi estabelecida a principal contribuição deste trabalho: auxiliar estudantes, investigadores, ou outros profissionais que necessitem de uma introdução ou uma visão geral sobre as linguagens disponíveis para a CAD, no momento de decisão sobre quais são as linguagens mais promissoras a explorar, num determinado contexto. Este EMS contribui para o corpo de conhecimento existente na área da CAD, consolidando a informação do estado da arte neste instante, oferecendo uma análise geral e individual sobre as linguagens utilizadas nesta área, com apontadores para estudos mais focados, respondendo às questões: quais são as categorias das linguagens utilizadas, qual é a natureza das mesmas, quais são os perfis típicos dos seus utilizadores, quão eficazes são essas linguagens e, por fim, quais são os tipos de artigos publicados nesta área.

5.2 Limitações

A criação deste EMS requereu uma extensa investigação bibliográfica. Esta tarefa implica um processo demorado, por envolver a análise e extração de dados de centenas de artigos. Apesar dos esforços tomados de modo a não ser excluída qualquer informação relevante sobre o tema, como a criação de uma *shortlist* de documentos que deveriam estar a ser encontrados nas bases de dados, tiveram de ser tomadas decisões como o filtrar das publicações na lista das conferências e revistas científicas consideradas as mais adequadas face ao tópico deste estudo secundário. Graças a tais medidas tomadas, tendo em conta o tempo disponível e os milhares de artigos existentes, existe uma probabilidade, ainda que reduzida, de não terem sido encontrados todos os estudos primários relevantes. Para além das limitações especificadas, uma pequena parte dos artigos adicionados na *shortlist* pedida aos especialistas não foi detetada com a investigação feita, mesmo sendo os documentos publicados por essas conferências ou revistas científicas, o que leva a concluir que as pesquisas automáticas não são totalmente confiáveis [Kit+10].

Outra limitação poderá ser a diversidade da terminologia utilizada pelos autores dos artigos publicados [Woh14b], pelo facto da mesma palavra poder ter interpretações diferentes e serem utilizados sinónimos das expressões habituais. Para tentar combater esses problemas, uma parte dos artigos foi analisada por um segundo revisor, de forma independente, e recorremos a variadas expressões com o mesmo significado para as consultas efetuadas nas plataformas eletrónicas. Em relação ao primeiro método, idealmente, a metodologia de realização deste tipo de estudos sugere que cada artigo seja revisto, de forma independente, por pelo menos duas pessoas. Os dados recolhidos são comparados e as discrepâncias são discutidas e resolvidas, eventualmente com o apoio de um terceiro revisor, mais sénior [KC07]. Assim, o risco de uma interpretação errada ficar registada é muito reduzido.

Por fim, é sabido que são criadas, constantemente, novas linguagens de programação, o que leva a que um estudo secundário como este nunca consiga ser totalmente completo nem atualizado, revelando-se, a médio prazo, uma outra limitação dos estudos secundários.

5.3 Trabalho Futuro

O trabalho futuro passa, principalmente, por promover a investigação e o aumento da quantidade e qualidade da literatura existente sobre o tópico das linguagens utilizadas para a CAD, no processamento de *Big Data*, não se focando apenas em reportar as bibliotecas, *Application Programming Interfaces (API)* e modelos de programação.

Por serem, continuamente, criadas novas linguagens de programação, este EMS deverá ser continuado e atualizado, a médio prazo, de modo a prosseguir com a sua contribuição para o corpo de conhecimento existente nesta área, consolidando a informação atual. O contexto deste estudo poderá ser estendido, no sentido de não só apresentar as linguagens utilizadas neste tipo de computação, como também as bibliotecas e restantes ferramentas integradas nas mesmas. Para tal, será necessário repetir os passos dados neste trabalho, recorrendo a uma nova procura, desta vez com diferentes palavras-chave, mais direcionadas para estas ferramentas de apoio, de modo a conseguir cobrir, novamente, a maior quantidade de informação possível. Para além do proposto, algumas sub-questões de pesquisa poderão ser estendidas. Um exemplo desta alteração seria, em relação à QI 4.2 - "Quais são os ganhos de produtividade/desempenho trazidos pelas linguagens reportadas e que tipo de medição foi utilizado?", investigar qual é o impacto desses ganhos.

Bibliografia

[Aboa] *About COST - cHiPSet ICT COST Action IC1406*. http://chipset-cost.eu/index.php/about-cost/. Acedido em: 19-01-2018. 2018.

[Aji+16] A. M. Aji, A. J. Peña, P. Balaji e W.-c. Feng. "MultiCL: Enabling automatic scheduling for task-parallel workloads in OpenCL". Em: *Parallel Computing* 58 (2016), pp. 37–55. DOI: https://doi.org/10.1016/j.parco.2016.05.006.

[Alb+15] R. Albodour, A. James e N. Yaacob. "QoS within business grid quality of service (BGQoS)". Em: *Future Generation Computer Systems* 50 (2015), pp. 22–37. DOI: https://doi.org/10.1016/j.future.2014.10.027.

[AD08] M. Aldinucci e M. Danelutto. "Securing skeletal systems with limited performance penalty: the Muskel experience". Em: *Journal of Systems Architecture* 54.9 (2008), pp. 868–876. DOI: https://doi.org/10.1016/j.sysarc.2008.02.008.

[Ald+17] M. Aldinucci, M. Danelutto, P. Kilpatrick, M. Torquati et al. "FastFlow: high-level and efficient streaming on multi-core". Em: *WILEY SERIES ON PARALLEL AND DISTRIBUTED COMPUTING* 86 (2017), pp. 261–280. DOI: https://doi.org/10.1002/9781119332015.ch13.

[Alh+11] N. K. Alham, M. Li, Y. Liu e S. Hammoud. "A MapReduce-based distributed SVM algorithm for automatic image annotation". Em: *Computers & Mathematics with Applications* 62.7 (2011), pp. 2801–2811. DOI: https://doi.org/10.1016/j.camwa.2011.07.046.

[Alm+14] S. S. de Almeida, A. C. de Nazaré Júnior, A. de Albuquerque Arauújo, G. Cámara-Chávez e D. Menotti. "Speeding up a video summarization approach using GPUs and multicore CPUs". Em: *Procedia Computer Science* 29 (2014), pp. 159–171. DOI: https://doi.org/10.1016/j.procs.2014.05.015.

[Anj+15] J. C. Anjos, I. Carrera, W. Kolberg, A. L. Tibola, L. B. Arantes e C. R. Geyer. "MRA++: Scheduling and data placement on MapReduce for heterogeneous environments". Em: *Future Generation Computer Systems* 42 (2015), pp. 22–35. DOI: https://doi.org/10.1016/j.future.2014.09.001.

[Had] *Apache Hadoop*. http://hadoop.apache.org/. Acedido em: 07-09-2018.

[Sto] *Apache Storm*. http://storm.apache.org/. Acedido em: 07-09-2018.

[AV15] M. Arif e H. Vandierendonck. "A Case Study of OpenMP Applied to Map/Reduce-Style Computations". Em: *International Workshop on OpenMP*. Springer. 2015, pp. 162–174.

[Infa] *Artigos de apoio Infopédia - análise qualitativa*. https://www.infopedia.pt/$analise-qualitativa. Acedido em: 16-01-2018. 2003.

[Infb] *Artigos de apoio Infopédia - análise quantitativa*. https://www.infopedia.pt/$analise-quantitativa. Acedido em: 16-01-2018. 2003.

[Bad+15] R. M. Badia, J. Conejero, C. Diaz, J. Ejarque, D. Lezzi, F. Lordan, C. Ramon-Cortes e R. Sirvent. "COMP superscalar, an interoperable programming framework". Em: *SoftwareX* 3 (2015), pp. 32–36. DOI: https://doi.org/10.1016/j.softx.2015.10.004.

[Bak12] K. Bakshi. "Considerations for Big Data: Architecture and Approach". Em: *Aerospace Conference, 2012 IEEE*. IEEE. 2012, pp. 1–7. ISBN: 978-1-4577-0557-1. DOI: https://doi.org/10.1109/AERO.2012.6187357.

[Bar+13] L. A. Barroso, J. Clidaras e U. Hölzle. "The datacenter as a computer: An introduction to the design of warehouse-scale machines". Em: *Synthesis lectures on computer architecture* 8.3 (2013), pp. 1–154. DOI: https://doi.org/10.2200/S00516ED2V01Y201306CAC024.

[Bar+10] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, P. Cornillon, M. Schildhauer, E. T. Borer, E. W. Seabloom et al. "Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis". Em: *Ecological Informatics* 5.1 (2010), pp. 42–50. DOI: https://doi.org/10.1016/j.ecoinf.2009.08.008.

[Bas+08] V. R. Basili, D. Cruzes, J. C. Carver, L. M. Hochstein, J. K. Hollingsworth, M. V. Zelkowitz e F. Shull. "Understanding the high-performance-computing community". Em: *IEEE software* 25.4 (2008), p29–36.

[Beh+16] R. Behrends, K. Hammond, V. Janjic, A. Konovalov, S. Linton, H.-W. Loidl, P. Maier e P. Trinder. "HPC-GAP: engineering a 21st-century high-performance computer algebra system". Em: *Concurrency and Computation: Practice and Experience* 28.13 (2016), pp. 3606–3636. DOI: https://doi.org/10.1002/cpe.3746.

[BC17] M. B. Belgacem e B. Chopard. "MUSCLE-HPC: A new high performance API to couple multiscale parallel applications". Em: *Future Generation Computer Systems* 67 (2017), pp. 72–82. DOI: https://doi.org/10.1016/j.future.2016.08.009.

[Ben+11] S. Benkner, S. Pllana, J. L. Traff, P. Tsigas, U. Dolinsky, C. Augonnet, B. Bachmayer, C. Kessler, D. Moloney e V. Osipov. "PEPPHER: Efficient and productive usage of hybrid computing systems". Em: *IEEE Micro* 31.5 (2011), pp. 28–41. DOI: http://doi.ieeecomputersociety.org/10.1109/MM.2011.670.

[BL12] M. A. Beyer e D. Laney. "The importance of big data: A definition". Em: *Stamford, CT: Gartner* (2012), pp. 2014–2018.

[Bin+13] A. P. D. Binotto, M. A. Wehrmeister, A. Kuijper e C. E. Pereira. "Sm@rtConfig: A context-aware runtime and tuning system using an aspect-oriented approach for data intensive engineering applications". Em: *Control Engineering Practice* 21.2 (2013), pp. 204–217. DOI: https://doi.org/10.1016/j.conengprac.2012.10.001.

[Bro+11] B. Brown, M. Chui e J. Manyika. "Are you Ready for the era of 'Big Data'". Em: *McKinsey Quarterly* 4.1 (2011), pp. 24–35.

[Bui+11] P. Bui, L. Yu, A. Thrasher, R. Carmichael, I. Lanc, P. Donnelly e D. Thain. "Scripting distributed scientific workflows using Weaver". Em: *Concurrency and Computation: Practice and Experience* 24.15 (2011), pp. 1685–1707. DOI: https://doi.org/10.1002/cpe.1871.

[Bur+16] L. Burgueño, M. Wimmer e A. Vallecillo. "A linda-based platform for the parallel execution of out-place model transformations". Em: *Information and Software Technology* 79 (2016), pp. 17–35. DOI: https://doi.org/10.1016/j.infsof.2016.06.001.

[BL15] M. Bux e U. Leser. "Dynamiccloudsim: Simulating heterogeneity in computational clouds". Em: *Future Generation Computer Systems* 46 (2015), pp. 85–99. DOI: https://doi.org/10.1016/j.future.2014.09.007.

[Buy+09] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg e I. Brandic. "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". Em: *Future Generation computer systems* 25.6 (2009), pp. 599–616. DOI: https://doi.org/10.1016/j.future.2008.12.001.

[Cab+15] R. Caballero, P. J. Stuckey e A. Tenorio-Fornes. "Two type extensions for the constraint modeling language MiniZinc". Em: *Science of Computer Programming* 111 (2015), pp. 156–189. DOI: https://doi.org/10.1016/j.scico.2015.04.007.

[Cal+16] J. Cala, E. Marei, Y. Xu, K. Takeda e P. Missier. "Scalable and efficient whole-exome data processing using workflows on the cloud". Em: *Future Generation Computer Systems* 65 (2016), pp. 153–168. DOI: https://doi.org/10.1016/j.future.2016.01.001.

[CG02] P. Cantos Gómez. "Do we need statistics when we have linguistics?" Em: *DELTA: Documentação de Estudos em Lingüística Teórica e Aplicada* 18.2 (2002), pp. 233–271. DOI: http://dx.doi.org/10.1590/S0102-44502002000200003.

[Car+13] G. Caruana, M. Li e Y. Liu. "An ontology enhanced parallel SVM for scalable spam filter training". Em: *Neurocomputing* 108 (2013), pp. 45–57. DOI: https://doi.org/10.1016/j.neucom.2012.12.001.

[Cav+14] S. Cavuoti, M. Garofalo, M. Brescia, M. Paolillo, A. Pescape, G. Longo e G. Ventre. "Astrophysical data mining with GPU. A case study: genetic classification of globular clusters". Em: *New Astronomy* 26 (2014), pp. 12–22. DOI: https://doi.org/10.1016/j.newast.2013.04.004.

[Cha+13] C. Chan, D. Unat, M. Lijewski, W. Zhang, J. Bell e J. Shalf. "Software design space exploration for exascale combustion co-design". Em: *International Supercomputing Conference*. Springer. 2013, pp. 196–212. DOI: https://doi.org/10.1007/978-3-642-38750-0_15.

[Che+15] L. Chen, X. Huo e G. Agrawal. "A pattern specification and optimizations framework for accelerating scientific computations on heterogeneous clusters". Em: *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*. IEEE. 2015, pp. 591–600. DOI: https://doi.org/10.1109/IPDPS.2015.13.

[CP16] A. Chianese e F. Picciali. "A service oriented framework for analysing social network activities". Em: *Procedia Computer Science* 98 (2016), pp. 509–514. DOI: https://doi.org/10.1016/j.procs.2016.09.087.

[CJ17] P Coetzee e S. Jarvis. "Goal-based composition of scalable hybrid analytics for heterogeneous architectures". Em: *Journal of Parallel and Distributed Computing* 108 (2017), pp. 59–73. DOI: https://doi.org/10.1016/j.jpdc.2016.11.009.

[Coe+14] P. Coetzee, M. Leeke e S Jarvis. "Towards unified secure on-and off-line analytics at scale". Em: *Parallel Computing* 40.10 (2014), pp. 738–753. DOI: https://doi.org/10.1016/j.parco.2014.07.004.

[Amd] *Compilers and More: Is Amdahl's Law Still Relevant?* https://www.hpcwire.com/2015/01/22/compilers-amdahls-law-still-relevant/. Acedido em: 05-06-2018.

[CV02] M. Coppola e M. Vanneschi. "High-performance data mining with skeleton-based structured parallel programming". Em: *Parallel Computing* 28.5 (2002), pp. 793–813. DOI: https://doi.org/10.1016/S0167-8191(02)00095-9.

[Cra15] S. Crafa. "The role of concurrency in an evolutionary view of programming abstractions". Em: *Journal of Logical and Algebraic Methods in Programming* 84.6 (2015), pp. 732–741. DOI: https://doi.org/10.1016/j.jlamp.2015.07.006.

[DM+15] A. De Mauro, M. Greco e M. Grimaldi. "What is big data? A consensual definition and a review of key research topics". Em: *AIP conference proceedings*. Vol. 1644. 1. AIP. 2015, pp. 97–104. DOI: https://doi.org/10.1063/1.4907823.

[DS+17] D. De Sensi, T. De Matteis, M. Torquati, G. Mencagli e M. Danelutto. "Bringing Parallel Patterns Out of the Corner: The P 3 ARSEC Benchmark Suite". Em: *ACM Transactions on Architecture and Code Optimization (TACO)* 14.1 (2017), p. 33. DOI: https://dl.acm.org/citation.cfm?doid=3132710.

[Ded+14] E. Dede, Z. Fadika, M. Govindaraju e L. Ramakrishnan. "Benchmarking MapReduce implementations under different application scenarios". Em: *Future Generation Computer Systems* 36 (2014), pp. 389–399. DOI: https://doi.org/10.1016/j.future.2014.01.001.

[Dee+09] E. Deelman, D. Gannon, M. Shields e I. Taylor. "Workflows and e-Science: An overview of workflow system features and capabilities". Em: *Future generation computer systems* 25.5 (2009), pp. 528–540. DOI: https://doi.org/10.1016/j.future.2008.06.012.

[Dee+15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny et al. "Pegasus, a workflow management system for science automation". Em: *Future Generation Computer Systems* 46 (2015), pp. 17–35. DOI: https://doi.org/10.1016/j.future.2014.10.008.

[DeV+11] Z. DeVito, N. Joubert, F. Palacios, S. Oakley, M. Medina, M. Barrientos, E. Elsen, F. Ham, A. Aiken, K. Duraisamy et al. "Liszt: a domain specific language for building portable mesh-based PDE solvers". Em: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2011, p. 9. DOI: https://doi.org/10.1145/2063384.2063396.

[Dia+15] P. D. Diamantoulakis, V. M. Kapinas e G. K. Karagiannidis. "Big data analytics for dynamic energy management in smart grids". Em: *Big Data Research* 2.3 (2015), pp. 94–101. DOI: https://doi.org/10.1016/j.bdr.2015.03.003.

[Dyb+05] T. Dyba, B. A. Kitchenham e M. Jorgensen. "Evidence-based software engineering for practitioners". Em: *IEEE Software* 22.1 (2005), pp. 58–65. ISSN: 0740-7459. DOI: https://doi.org/10.1109/MS.2005.6.

[Edm+13] N. Edmonds, J. Willcock e A. Lumsdaine. "Expressing graph algorithms using generalized active messages". Em: *Proceedings of the 27th international ACM conference on International conference on supercomputing*. ACM. 2013, pp. 283–292. DOI: https://doi.org/10.1145/2464996.2465441.

[EP16] N. Emad e S. Petiton. "Unite and conquer approach for high scale numerical computing". Em: *Journal of computational science* 14 (2016), pp. 5–14. DOI: https://doi.org/10.1016/j.jocs.2016.01.007.

[EK10] J. Enmyren e C. W. Kessler. "SkePU: a Multi-Backend Skeleton Programming Library for Multi-GPU Systems". Em: *Proceedings of the fourth international workshop on High-level parallel programming and applications*. ACM. 2010, pp. 5–14. DOI: https://doi.org/10.1145/1863482.1863487.

[Erä+14] D. Eränen, J. Oksanen, J. Westerholm e T. Sarjakoski. "A full graphics processing unit implementation of uncertainty-aware drainage basin delineation". Em: *Computers & Geosciences* 73 (2014), pp. 48–60. DOI: https://doi.org/10.1016/j.cageo.2014.08.012.

[Err+15] U. Erra, S. Senatore, F. Minnella e G. Caggianese. "Approximate TF–IDF based on topic extraction from massive message stream using the GPU". Em: *Information Sciences* 292 (2015), pp. 143–161. DOI: https://doi.org/10.1016/j.ins.2014.08.062.

[EO+16] J. Espinosa-Oviedo, G. Vargas-Solar, V. Alexandrov e G. Castel. "Comparing Electoral Campaigns by Analysing Online Data". Em: *Procedia Computer Science* 80 (2016), pp. 1865–1874. DOI: https://doi.org/10.1016/j.procs.2016.05.480.

[Fal+14] Z. Falt, D. Bednárek, M. Krulis, J. Yaghob e F. Zavoral. "Bobolang: A language for parallel streaming applications". Em: *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing*. ACM. 2014, pp. 311–314. DOI: https://doi.org/10.1145/2600212.2600711.

[Fer+14] H. Fernandez, C. Tedeschi e T. Priol. "Rule-driven service coordination middleware for scientific applications". Em: *Future Generation Computer Systems* 35 (2014), pp. 1–13. DOI: https://doi.org/10.1016/j.future.2013.12.023.

[Fer+16] M. Fernández, J. C. Pichel, J. C. Cabaleiro e T. F. Pena. "Boosting performance of a statistical machine translation system using dynamic parallelism". Em: *Journal of Computational Science* 13 (2016), pp. 37–48. DOI: https://doi.org/10.1016/j.jocs.2016.01.003.

[Fer+15] V. Fernández, V. Méndez e T. F. Pena. "Federated Big Data for resource aggregation and load balancing with DIRAC". Em: *Procedia Computer Science* 51 (2015), pp. 2769–2773. DOI: https://doi.org/10.1016/j.procs.2015.05.430.

[Fol+10] G. Folino, A. Forestiero, G. Papuzzo e G. Spezzano. “A grid portal for solving geoscience problems using distributed knowledge discovery services”. Em: *Future Generation Computer Systems* 26.1 (2010), pp. 87–96. DOI: https://doi.org/10.1016/j.future.2009.08.002.

[For+14] G. Fortino, D. Parisi, V. Pirrone e G. Di Fatta. “BodyCloud: A SaaS approach for community body sensor networks”. Em: *Future Generation Computer Systems* 35 (2014), pp. 62–79. DOI: https://doi.org/10.1016/j.future.2013.12.015.

[Gab+10] P. Gabriel, M. Goulão e V. Amaral. “Do Software Languages Engineers Evaluate their Languages?” Em: *XIII Congreso Iberoamericano en "Software Engineering"(CIbSE'2010), Cuenca, Ecuador* (2010).

[GN00] E. R. Gansner e S. C. North. “An open graph visualization system and its applications to software engineering”. Em: *Software Practice and Experience* 30.11 (2000), pp. 1203–1233.

[Gar+08] V. Garousi, L. C. Briand e Y. Labiche. “Traffic-aware stress testing of distributed real-time systems based on UML models using genetic algorithms”. Em: *Journal of Systems and Software* 81.2 (2008), pp. 161–185. DOI: https://doi.org/10.1016/j.jss.2007.05.037.

[GH07] T. Gautier e H.-R. Hamidi. “Re-scheduling invocations of services for RPC grids”. Em: *Computer Languages, Systems & Structures* 33.3-4 (2007), pp. 168–178. DOI: https://doi.org/10.1016/j.cl.2006.07.006.

[Gia15] R. Giachetta. “A framework for processing large scale geospatial and remote sensing data in MapReduce environment”. Em: *Computers & graphics* 49 (2015), pp. 37–46. DOI: https://doi.org/10.1016/j.cag.2015.03.003.

[Gil+13] M. B. Giles, G. R. Mudalige, B Spencer, C. Bertolli e I Reguly. “Designing OP2 for GPU architectures”. Em: *Journal of Parallel and Distributed Computing* 73.11 (2013), pp. 1451–1460. DOI: https://doi.org/10.1016/j.jpdc.2012.07.008.

[Giu+11] G. Giuliani, N. Ray e A. Lehmann. “Grid-enabled Spatial Data Infrastructure for environmental sciences: Challenges and opportunities”. Em: *Future Generation Computer Systems* 27.3 (2011), pp. 292–303. DOI: https://doi.org/10.1016/j.future.2010.09.011.

[Gon+15] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang e J.-J. Li. “Distributed evolutionary algorithms and their models: A survey of the state-of-the-art”. Em: *Applied Soft Computing* 34 (2015), pp. 286–300. DOI: https://doi.org/10.1016/j.asoc.2015.04.061.

[Gra+15] I. Grasso, M. Ritter, B. Cosenza, W. Benger, G. Hofstetter e T. Fahringer. "Point distribution tensor computation on heterogeneous systems". Em: *Procedia Computer Science* 51 (2015), pp. 160–169. DOI: https://doi.org/10.1016/j.procs.2015.05.217.

[Gub+13] J. Gubbi, R. Buyya, S. Marusic e M. Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions". Em: *Future generation computer systems* 29.7 (2013), pp. 1645–1660. DOI: https://doi.org/10.1016/j.future.2013.01.010.

[Guh+09] R. Guha, N. Bagherzadeh e P. Chou. "Resource management and task partitioning and scheduling on a run-time reconfigurable embedded system". Em: *Computers & Electrical Engineering* 35.2 (2009), pp. 258–285. DOI: https://doi.org/10.1016/j.compeleceng.2008.06.008.

[Gun+13] T. Gunarathne, B. Zhang, T.-L. Wu e J. Qiu. "Scalable parallel computing on clouds using Twister4Azure iterative MapReduce". Em: *Future Generation Computer Systems* 29.4 (2013), pp. 1035–1048. DOI: https://doi.org/10.1016/j.future.2012.05.027.

[HN15a] M. Hammoudeh e R. Newman. "Information extraction from sensor networks using the Watershed transform algorithm". Em: *Information Fusion* 22 (2015), pp. 39–49. DOI: https://doi.org/10.1016/j.inffus.2013.07.001.

[Han+11] L. Han, C. S. Liew, J. Van Hemert e M. Atkinson. "A generic parallel processing model for facilitating data mining and integration". Em: *Parallel Computing* 37.3 (2011), pp. 157–171. DOI: https://doi.org/10.1016/j.parco.2011.02.0060.

[Han+15] R. Hanisch, G. Berriman, T. Lazio, S. E. Bunn, J Evans, T. McGlynn e R Plante. "The Virtual Astronomical Observatory: Re-engineering access to astronomical data". Em: *Astronomy and Computing* 11 (2015), pp. 190–209. DOI: https://doi.org/10.1016/j.ascom.2015.03.007.

[He+16] K. He, S. X.-D. Tan, H. Zhao, X.-X. Liu, H. Wang e G. Shi. "Parallel GMRES solver for fast analysis of large linear dynamic systems on GPU platforms". Em: *INTEGRATION, the VLSI journal* 52 (2016), pp. 10–22. DOI: https://doi.org/10.1016/j.vlsi.2015.07.005.

[Hij+15] P. Hijma, R. Van Nieuwpoort, C. J. Jacobs e H. E. Bal. "Stepwise-refinement for performance: a methodology for many-core programming". Em: *Concurrency and Computation: Practice and Experience* 27.17 (2015), pp. 4515–4554. DOI: https://doi.org/10.1002/cpe.3416.

[Hin+06] K. Hinsen, H. P. Langtangen, O. Skavhaug e Å. Ødegård. "Using B SP and Python to simplify parallel programming". Em: *Future Generation Computer Systems* 22.1-2 (2006), pp. 123–157. DOI: https://doi.org/10.1016/j.future.2003.09.003.

[How] *How to use high-performance computing to analyze big data*. http://searchmicroservices.techtarget.com/tip/How-to-use-high-performance-computing-to-analyze-big-data. Acedido em: 15-01-2018.

[Hsu14] C.-H. Hsu. "Intelligent big data processing". Em: *Future Generation Computer Systems* 36 (2014), pp. 16–18. DOI: https://doi.org/10.1016/j.future.2014.02.003.

[Hsu+15] C.-H. Hsu, K. D. Slagter e Y.-C. Chung. "Locality and loading aware virtual machine mapping techniques for optimizing communications in MapReduce applications". Em: *Future Generation Computer Systems* 53 (2015), pp. 43–54. DOI: https://doi.org/10.1016/j.future.2015.04.006.

[HN15b] L. Hu e S. Nooshabadi. "G-SHOT: GPU accelerated 3D local descriptor for surface matching". Em: *Journal of Visual Communication and Image Representation* 30 (2015), pp. 343–349. DOI: https://doi.org/10.1016/j.jvcir.2015.05.008.

[Hua+17] F. Huang, J. Tao, Y. Xiang, P. Liu, L. Dong e L. Wang. "Parallel compressive sampling matching pursuit algorithm for compressed sensing signal reconstruction with OpenCL". Em: *Journal of Systems Architecture* 72 (2017), pp. 51–60. DOI: https://doi.org/10.1016/j.sysarc.2016.07.002.

[Hün+15] D. Hünich, A. Knüpfer e J. Gracia. "Providing Parallel Debugging for DASH Distributed Data Structures with GDB". Em: *Procedia Computer Science* 51 (2015), pp. 1383–1392. DOI: https://doi.org/10.1016/j.procs.2015.05.345.

[Jin+17] W. Jing, D. Tong, Y. Wang, J. Wang, Y. Liu e P. Zhao. "MaMR: High-performance MapReduce programming model for material cloud applications". Em: *Computer Physics Communications* 211 (2017), pp. 79–87. DOI: https://doi.org/10.1016/j.cpc.2016.07.015.

[Jor+05] M. Jorgensen, T. Dyba e B. Kitchenham. "Teaching Evidence-Based Software Engineering to University Students". Em: *IEEE Software* (2005).

[KP13] C. Kaiser e A. Pozdnoukhov. "Enabling real-time city sensing with kernel stream oracles and MapReduce". Em: *Pervasive and Mobile Computing* 9.5 (2013), pp. 708–721. DOI: https://doi.org/10.1016/j.pmcj.2012.11.003.

[Kaj+14] T. Kajdanowicz, P. Kazienko e W. Indyk. "Parallel processing of large graphs". Em: *Future Generation Computer Systems* 32 (2014), pp. 324–337. DOI: https://doi.org/10.1016/j.future.2013.08.007.

[Kal13] R. S. Kalawsky. "The Next Generation of Grand Challenges for Systems Engineering Research". Em: *Procedia Computer Science* 16 (2013), pp. 834–843. DOI: https://doi.org/10.1016/j.procs.2013.01.087.

[Kal+16] E. Kaldeli, A. Lazovik e M. Aiello. "Domain-independent planning for services in uncertain and dynamic environments". Em: *Artificial Intelligence* 236 (2016), pp. 30–64. DOI: https://doi.org/10.1016/j.artint.2016.03.002.

[Kat+16] D. S. Katz, A. Merzky, Z. Zhang e S. Jha. "Application skeletons: Construction and use in eScience". Em: *Future Generation Computer Systems* 59 (2016), pp. 114–124. DOI: https://doi.org/10.1016/j.future.2015.10.001.

[Kim+15] M. Kim, Y. Lee, H.-H. Park, S. J. Hahn e C.-G. Lee. "Computational fluid dynamics simulation based on Hadoop Ecosystem and heterogeneous computing". Em: *Computers & Fluids* 115 (2015), pp. 1–10. DOI: https://doi.org/10.1016/j.compfluid.2015.03.021.

[KC07] B. Kitchenham e S. Charters. "Guidelines for performing Systematic Literature Reviews in Software Engineering". Em: *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. sn, 2007, pp. 1–57.

[Kit+09] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey e S. Linkman. "Systematic literature reviews in software engineering – A systematic literature review". Em: *Information and Software Technology* 51.1 (2009), 7––15. DOI: https://doi.org/10.1016/j.infsof.2008.09.009.

[Kit+10] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi e S. Linkman. "Systematic literature reviews in software engineering – A tertiary study". Em: *Information and Software Technology* 52.8 (2010), 792––805. DOI: https://doi.org/10.1016/j.infsof.2010.03.006.

[Kit+02] B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam e J. Rosenberg. "Preliminary Guidelines for Empirical Research in Software Engineering". Em: *IEEE Transactions on Software Engineering* 28.8 (2002), pp. 721–734. DOI: https://doi.org/10.1109/TSE.2002.1027796.

[Kit+04] B. A. Kitchenham, T. Dyba e M. Jorgensen. "Evidence-Based Software Engineering". Em: *Proceedings of the 26th international conference on software engineering*. IEEE Computer Society. 2004, pp. 273–281. ISBN: 0-7695-2163-0.

[Kol+13] W. Kolberg, P. D. B. Marcos, J. C. Anjos, A. K. Miyazaki, C. R. Geyer e L. B. Arantes. "Mrsg–a mapreduce simulator over simgrid". Em: *Parallel Computing* 39.4-5 (2013), pp. 233–244. DOI: https://doi.org/10.1016/j.parco.2013.02.001.

[Kon+11] R. Koning, P. Grosso e C. de Laat. "Using ontologies for resource description in the CineGrid Exchange". Em: *Future Generation Computer Systems* 27.7 (2011), pp. 960–965. DOI: https://doi.org/10.1016/j.future.2010.11.027.

[Kos+10] T. Kosar, N. Oliveira, M. Mernik, M. J. V. Pereira, M. Črepinšek, D. da Cruz e P. R. Henriques. "Comparing General-Purpose and Domain-Specific Languages: An Empirical Study". Em: *Computer Science and Information Systems* 7.2 (2010), pp. 247–264. DOI: https://doi.org/10.2298/CSIS1002247K.

[Kos+12] T. Kosar, M. Mernik e J. C. Carver. "Program comprehension of domain-specific and general-purpose languages: comparison using a family of experiments". Em: *Empirical Software Engineering* 17.3 (2012), pp. 276–304. ISSN: 1573-7616. DOI: https://doi.org/10.1007/s10664-011-9172-x.

[Kos+16] T. Kosar, S. Bohra e M. Mernik. "Domain-Specific Languages: A Systematic Mapping Study". Em: *Information and Software Technology* 71 (2016), pp. 77–91. DOI: https://doi.org/10.1016/j.infsof.2015.11.001.

[KS15] M. Krämer e I. Senner. "A modular software architecture for processing of big geospatial data in the cloud". Em: *Computers & Graphics* 49 (2015), pp. 69–81. DOI: https://doi.org/10.1016/j.cag.2015.02.005.

[Kra+15] J. Kranjc, J. Smailović, V. Podpečan, M. Grčar, M. Žnidaršič e N. Lavrač. "Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the ClowdFlows platform". Em: *Information Processing & Management* 51.2 (2015), pp. 187–203. DOI: https://doi.org/10.1016/j.ipm.2014.04.001.

[Kra+17] J. Kranjc, R. Orač, V. Podpečan, N. Lavrač e M. Robnik-Šikonja. "ClowdFlows: Online workflows for distributed big data mining". Em: *Future Generation Computer Systems* 68 (2017), pp. 38–58. DOI: https://doi.org/10.1016/j.future.2016.07.018.

[LJ12] A. Labrinidis e H. V. Jagadish. "Challenges and Opportunities with Big Data". Em: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 2032–2033. ISSN: 2150-8097. DOI: http://doi.acm.org/10.14778/2367502.2367572.

[Lee+09] H.-G. Lee, N. Park, H.-I. Jeong e J. Park. "Grid enabled MRP process improvement under distributed database environment". Em: *Journal of Systems and Software* 82.7 (2009), pp. 1087–1097. DOI: https://doi.org/10.1016/j.jss.2009.01.041.

[Li+15] X. Li, M. Grossman e D. Kaeli. "Mahout on heterogeneous clusters using HadoopCL". Em: *Proceedings of the 2nd Workshop on Parallel Programming for Analytics Applications*. ACM. 2015, pp. 9–16. DOI: https://doi.org/10.1145/2726935.2726940.

[Lia+07] T.-Y. Liang, C.-Y. Wu, C.-K. Shieh e J.-B. Chang. "A grid-enabled software distributed shared memory system on a wide area network". Em: *Future Generation Computer Systems* 23.4 (2007), pp. 547–557. DOI: https://doi.org/10.1016/j.future.2006.10.003.

[Lia+16] T.-Y. Liang, H.-F. Li, Y.-J. Lin e B.-S. Chen. "A Distributed PTX Virtual Machine on Hybrid CPU/GPU Clusters". Em: *Journal of Systems Architecture* 62 (2016), pp. 63–77. DOI: https://doi.org/10.1016/j.sysarc.2015.10.003.

[LS09] C. Lin e L. Snyder. "Principles of Parallel Programming". Em: Pearson Int Ed., 2009, pp. 1–29. ISBN: 0-3214-8790-7.

[LD10] J. Lin e C. Dyer. "Data-intensive text processing with MapReduce". Em: *Synthesis Lectures on Human Language Technologies* 3.1 (2010), pp. 1–177. DOI: https://doi.org/10.2200/S00274ED1V01Y201006HLT007.

[LG+16] M. Liroz-Gistau, R. Akbarinia, D. Agrawal e P. Valduriez. "FP-Hadoop: Efficient processing of skewed MapReduce jobs". Em: *Information Systems* 60 (2016), pp. 69–84. DOI: https://doi.org/10.1016/j.is.2016.03.008.

[Liu+15] G. Liu, W. Zhu, C. Saunders, F. Gao e Y. Yu. "Real-time complex event processing and analytics for smart grid". Em: *Procedia Computer Science* 61 (2015), pp. 113–119. DOI: https://doi.org/10.1016/j.procs.2015.09.169.

[Luc+15] A. Luckow, M. Santcroos, A. Zebrowski e S. Jha. "Pilot-data: an abstraction for distributed data". Em: *Journal of Parallel and Distributed Computing* 79-80 (2015), pp. 16–30. DOI: https://doi.org/10.1016/j.jpdc.2014.09.009.

[Lug+15] A. Lugowski, S. Kamil, A. Buluç, S. Williams, E. Duriakova, L. Oliker, A. Fox e J. R. Gilbert. "Parallel processing of filtered queries in attributed semantic graphs". Em: *Journal of Parallel and Distributed Computing* 79 (2015), pp. 115–131. DOI: https://doi.org/10.1016/j.jpdc.2014.08.010.

[Ma+15] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya e W. Jie. "Remote sensing big data computing: Challenges and opportunities". Em: *Future Generation Computer Systems* 51 (2015), pp. 47–60. DOI: https://doi.org/10.1016/j.future.2014.10.029.

[Ma+14] Z. Ma, H Wang e S. Pu. "GPU computing of compressible flow problems by a meshless method with space-filling curves". Em: *Journal of Computational Physics* 263 (2014), pp. 113–135. DOI: https://doi.org/10.1016/j.jcp.2014.01.023.

[Mad12] S. Madden. "From databases to big data". Em: *IEEE Internet Computing* 16.3 (2012), pp. 4–6. DOI: https://doi.org/10.1109/MIC.2012.50.

[Mah+16] K. Maheshwari, E.-S. Jung, J. Meng, V. Morozov, V. Vishwanath e R. Kettimuthu. "Workflow performance improvement using model-based scheduling over multiple clusters and clouds". Em: *Future Generation Computer Systems* 54 (2016), pp. 206–218. DOI: https://doi.org/10.1016/j.future.2015.03.017.

[Mal16] R. Malhotra. "Empirical research in software engineering: concepts, analysis, and applications". Em: CRC Press, 2016, pp. 1–61. ISBN: 978-1-4987-1973-5.

[Map] *MapReduce Tutorial | Mapreduce Example in Apache Hadoop | Edureka*. https://www.edureka.co/blog/mapreduce-tutorial/. Acedido em: 20-11-2018.

[Mar13] D. C. Marinescu. *Cloud Computing: Theory and Practice*. 1st Edition. Morgan Kaufman-Elsevier, 2013. ISBN: 9780124046412. DOI: https://www.elsevier.com/books/cloud-computing/marinescu/978-0-12-404627-6.

[Mat+10] C. Mateos, A. Zunino e M. Campo. "An approach for non-intrusively adding malleable fork/join parallelism into ordinary JavaBean compliant applications". Em: *Computer Languages, Systems & Structures* 36.3 (2010), pp. 288–315. DOI: https://doi.org/10.1016/j.cl.2009.12.003.

[Mat+11] C. Mateos, A. Zunino, M. Hirsch, M. Fernández e M. Campo. "A software tool for semi-automatic gridification of resource-intensive Java bytecodes and its application to ray tracing and sequence alignment". Em: *Advances in Engineering Software* 42.4 (2011), pp. 172–186. DOI: https://doi.org/10.1016/j.advengsoft.2011.02.003.

[May+11] C. Mayr, U. Zdun e S. Dustdar. "View-based model-driven architecture for enhancing maintainability of data access services". Em: *Data & Knowledge Engineering* 70.9 (2011), pp. 794–819. DOI: https://doi.org/10.1016/j.datak.2011.05.004.

[McC+07] P. McCormick, J. Inman, J. Ahrens, J. Mohd-Yusof, G. Roth e S. Cummins. "Scout: a data-parallel programming language for graphics processors". Em: *Parallel Computing* 33.10-11 (2007), pp. 648–662. DOI: https://doi.org/10.1016/j.parco.2007.09.001.

[Mea+17] A. Meade, D. K. Deeptimahanti, J. Buckley e J. Collins. "An empirical study of data decomposition for software parallelization". Em: *Journal of Systems and Software* 125 (2017), pp. 401–416. DOI: https://doi.org/10.1016/j.jss.2016.02.002.

[Mem+17] S. Memeti, L. Li, S. Pllana, J. Kołodziej e C. Kessler. "Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: programming productivity, performance, and energy consumption". Em: *Proceedings of the 2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*. ACM. 2017, pp. 1–6. DOI: https://doi.org/10.1145/3110355.3110356.

[Men+17] G. Mencagli, M. Torquati, F. Lucattini, S. Cuomo e M. Aldinucci. "Harnessing sliding-window execution semantics for parallel stream processing". Em: *Journal of Parallel and Distributed Computing* (2017). DOI: https://doi.org/10.1016/j.jpdc.2017.10.021.

[Mic+17] É. Michon, J. Gossa, S. Genaud, L. Unbekandt e V. Kherbache. "Schlouder: A broker for IaaS clouds". Em: *Future Generation Computer Systems* 69 (2017), pp. 11–23. DOI: https://doi.org/10.1016/j.future.2016.09.010.

[Mis+17] C. Misale, M. Drocco, M. Aldinucci e G. Tremblay. "A comparison of big data frameworks on a layered dataflow model". Em: *Parallel Processing Letters* 27.01 (2017), p. 1740003. DOI: https://doi.org/10.1142/S0129626417400035.

[Mis+18] C. Misale, M. Drocco, G. Tremblay, A. R. Martinelli e M. Aldinucci. "PiCo: High-performance data analytics pipelines in modern C++". Em: *Future Generation Computer Systems* 87 (2018), pp. 392–403. DOI: https://doi.org/10.1016/j.future.2018.05.030.

[Mor+07] A. Morajko, T. Margalef e E. Luque. "Design and implementation of a dynamic tuning environment". Em: *Journal of Parallel and Distributed Computing* 67.4 (2007), pp. 474–490. DOI: https://doi.org/10.1016/j.jpdc.2007.01.001.

[Mou+14] D. Mourtzis, M. Doukas e D. Bernidaki. "Simulation in manufacturing: Review and challenges". Em: *Procedia CIRP* 25 (2014), pp. 213–229. DOI: https://doi.org/10.1016/j.procir.2014.10.032.

[Abob] *NOVA-LINCS - HISTORY*. http://nova-lincs.di.fct.unl.pt/the-center/history. Acedido em: 19-01-2018. 2018.

[Núñ+12] A. Núñez, J. Fernández, R. Filgueira, F. García e J. Carretero. "SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications". Em: *Simulation Modelling Practice and Theory* 20.1 (2012), pp. 12–32. DOI: https://doi.org/10.1016/j.simpat.2011.08.009.

[Obr+11] C. Obrecht, F. Kuznik, B. Tourancheau e J.-J. Roux. "A new approach to the lattice Boltzmann method for graphics processing units". Em: *Computers & Mathematics with Applications* 61.12 (2011), pp. 3628–3638. DOI: https://doi.org/10.1016/j.camwa.2010.01.054.

[Obr+12] C. Obrecht, F. Kuznik, B. Tourancheau e J.-J. Roux. "The TheLMA project: A thermal lattice Boltzmann solver for the GPU". Em: *Computers & Fluids* 54 (2012), pp. 118–126. DOI: https://doi.org/10.1016/j.compfluid.2011.10.011.

[Oos+15] P. van Oosterom, O. Martinez-Rubi, M. Ivanova, M. Horhammer, D. Geringer, S. Ravada, T. Tijssen, M. Kodde e R. Gonçalves. "Massive point cloud data management: Design, implementation and execution of a point cloud benchmark". Em: *Computers & Graphics* 49 (2015), pp. 92–125. DOI: https://doi.org/10.1016/j.cag.2015.01.007.

[Pac11] P. Pacheco. "An Introduction to Parallel Programming". Em: Morgan Kaufmann Publishers, 2011, pp. 1–14. ISBN: 978-0-12-374260-5.

[Pan+17] A. Panyala, D. Chavarría-Miranda, J. B. Manzano, A. Tumeo e M. Halappanavar. "Exploring performance and energy tradeoffs for irregular applications: A case study on the Tilera many-core architecture". Em: *Journal of Parallel and Distributed Computing* 104 (2017), pp. 234–251. DOI: https://doi.org/10.1016/j.jpdc.2016.06.006.

[PN14] A. Papagiannis e D. S. Nikolopoulos. "Hybrid address spaces: A methodology for implementing scalable high-level programming models on non-coherent many-core architectures". Em: *Journal of Systems and Software* 97 (2014), pp. 47–64. DOI: https://doi.org/10.1016/j.jss.2014.06.058.

[Pap+17] G. A. Papakostas, K. I. Diamantaras e T. Papadimitriou. "Parallel pattern classification utilizing GPU-based kernelized Slackmin algorithm". Em: *Journal of Parallel and Distributed Computing* 99 (2017), pp. 90–99. DOI: https://doi.org/10.1016/j.jpdc.2016.09.001.

[Pet+08a] K. Petersen, R. Feldt, S. Mujtaba e M. Mattsson. "Systematic Mapping Studies in Software Engineering". Em: *Evaluation and Assessment in Software Engineering*. Vol. 8. 2008, pp. 68–77.

[Pet+08b] K. Petersen, R. Feldt, S. Mujtaba e M. Mattsson. "Systematic Mapping Studies in Software Engineering." Em: *EASE*. Vol. 8. 2008, pp. 68–77. DOI: https://www.researchgate.net/publication/228350426.

[PR05] M. Petticrew e H. Roberts. *Systematic Reviews in the Social Sciences: A Practical Guide*. Blackwell Publishing, 2005. ISBN: 978-1-405-12110-1.

[PA09] S. L. Pfleeger e J. M. Atlee. *Software engineering - Theory and Practice*. Pearson, 2009. ISBN: 0136061699.

[Pic+15] B. P. Pickering, C. W. Jackson, T. R. W. Scogland, W.-C. Feng e C. J. Roy. "Directive-based GPU programming for computational fluid dynamics". Em: *Computers & Fluids* 114 (2015), pp. 242–253. DOI: https://doi.org/10.1016/j.compfluid.2015.03.008.

[Pin+15] L. M. Pinho, V. Nélis, P. M. Yomsi, E. Quiñones, M. Bertogna, P. Burgio, A. Marongiu, C. Scordino, P. Gai, M. Ramponi et al. "P-SOCRATES: A parallel software framework for time-critical many-core systems". Em: *Microprocessors and Microsystems* 39.8 (2015), pp. 1190–1203. DOI: https://doi.org/10.1016/j.micpro.2015.06.004.

[PM14] R. S. Pressman e B. Maxim. *Software engineering: a Practitioner's Approach*. 8th edition. McGraw-Hill, 2014. ISBN: 0078022126.

[Rat16] S. Rathi. "Optimizing Sorting Algorithms using Ubiquitous multi-core massively parallel GPGPU processors". Em: *Procedia Computer Science* 79 (2016), pp. 231–237. DOI: https://doi.org/10.1016/j.procs.2016.03.030.

[RD09] C. for Reviews e Dissemination. *Systematic Reviews: CRD's Guidance for Undertaking Reviews in Healthcare*. CRD, University of York, 2009, pp. 1–281. ISBN: 978-1-900640-47-3.

[Ros+16] J. A. Ross, D. A. Richie, S. J. Park e D. R. Shires. "Parallel programming model for the Epiphany many-core coprocessor using threaded MPI". Em: *Microprocessors and Microsystems* 43 (2016), pp. 95–103. DOI: https://doi.org/10.1016/j.micpro.2016.02.006.

[Sac+00] D. L. Sackett, S. E. Straus, W. Richardson, W. Rosenberg e R. B. Haynes. *Evidence-Based Medicine: How to Practice and Teach EBM*. 2nd Edition. Churchill Livingstone, 2000. ISBN: 9780702031274.

[SS13] S. Sagiroglu e D. Sinanc. "Big Data: A Review". Em: *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE, 2013, pp. 42–47. ISBN: 978-1-4673-6404-1. DOI: https://doi.org/10.1109/CTS.2013.6567202.

[Sal+16] L. Salucci, D. Bonetta e W. Binder. "Lightweight Multi-language Bindings for Apache Spark". Em: *European Conference on Parallel Processing*. Springer. 2016, pp. 281–292. DOI: https://doi.org/10.1007/978-3-319-43659-3_21.

[Sat15] I. Satoh. "Pervasive Data Processing". Em: *Procedia Computer Science* 63 (2015), pp. 16–23. DOI: https://doi.org/10.1016/j.procs.2015.08.307.

[Jav] *Satshabad/Simple-Graph-Implementation*. https://github.com/Satshabad/Simple-Graph-Implementation/blob/master/src/Graph.java. Acedido em: 04-08-2018.

[Sch+11] M. Schneider, D. Fey, D. Kapusi e T. Machleidt. "Performance comparison of designated preprocessing white light interferometry algorithms on emerging multi-and many-core architectures". Em: *Procedia Computer Science* 4 (2011), pp. 2037–2046. DOI: https://doi.org/10.1016/j.procs.2011.04.222.

[Sch+15] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann et al. "T-CREST: Time-predictable multi-core architecture for embedded systems". Em: *Journal of Systems Architecture* 61.9 (2015), pp. 449–471. DOI: https://doi.org/10.1016/j.sysarc.2015.04.002.

[Sea99] C. B. Seaman. "Qualitative methods in empirical studies of software engineering". Em: *IEEE Transactions on software engineering* 25.4 (1999), pp. 557–572.

[Sec+17] C. Seceleanu, M. Johansson, J. Suryadevara, G. Sapienza, T. Seceleanu, S.-E. Ellevseth e P. Pettersson. "Analyzing a wind turbine system: From simulation to formal verification". Em: *Science of Computer Programming* 133 (2017), pp. 216–242. DOI: https://doi.org/10.1016/j.scico.2016.09.007.

[Sec+10] J. Secretan, M. Georgiopoulos, A. Koufakou e K. Cardona. "APHID: An architecture for private, high-performance integrated data mining". Em: *Future Generation Computer Systems* 26.7 (2010), pp. 891–904. DOI: https://doi.org/10.1016/j.future.2010.02.017.

[SR14] M. R. Selim e M. Z. Rahman. "Carrying on the legacy of imperative languages in the future parallel computing era". Em: *Parallel Computing* 40.3-4 (2014), pp. 1–33. DOI: https://doi.org/10.1016/j.parco.2014.02.001.

[Sen+15] D. Sengupta, S. L. Song, K. Agarwal e K. Schwan. "GraphReduce: processing large-scale graphs on accelerator-based systems". Em: *High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for*. IEEE. 2015, pp. 1–12. DOI: https://doi.org/10.1145/2807591.2807655.

[Ser+16] D. Serrano, S. Bouchenak, Y. Kouki, F. A. de Oliveira Jr, T. Ledoux, J. Lejeune, J. Sopena, L. Arantes e P. Sens. "SLA guarantees for cloud services". Em: *Future Generation Computer Systems* 54 (2016), pp. 233–246. DOI: https://doi.org/10.1016/j.future.2015.03.018.

[Sim+15] A. Simonet, G. Fedak e M. Ripeanu. "Active Data: A programming model to manage data life cycle across heterogeneous systems and infrastructures". Em: *Future Generation Computer Systems* 53 (2015), pp. 25–42. DOI: https://doi.org/10.1016/j.future.2015.05.015.

[SB16] H. Singh e S. Bawa. "HadoopWeb: MapReduce Platform for Big Data Analysis". Em: *International Research Journal of Engineering and Technology (IRJET)* 3.7 (2016), pp. 1355–1361.

[SS12] S. Singh e N. Singh. "Big Data Analytics". Em: *2012 International Conference on Communication, Information and Computing Technology* (2012).

[Siv+17] U. Sivarajah, M. M. Kamal, Z. Irani e V. Weerakkody. "Critical analysis of Big Data challenges and analytical methods". Em: *Journal of Business Research* 70 (2017), pp. 263–286. DOI: https://doi.org/10.1016/j.jbusres.2016.08.001.

[Sjö+16] O. Sjöström, S.-H. Ko, U. Dastgeer, L. Li e C. Kessler. "Portable parallelization of the EDGE CFD application for GPU-based systems using the SkePU skeleton programming library". Em: *ParCo-2015 conference* 27 (2016), pp. 135–144. DOI: https://doi.org/10.3233/978-1-61499-621-7-135.

[Som15] I. Sommerville. *Software Engineering*. 10th edition. Pearson, 2015. ISBN: 9781292096131.

[Son+12] K. Soni, D. D. Chandar e J. Sitaraman. "Development of an overset grid computational fluid dynamics solver on graphical processing units". Em: *Computers & Fluids* 58 (2012), pp. 1–14. DOI: https://doi.org/10.1016/j.compfluid.2011.11.014.

[Su+11] Y.-L. Su, P.-C. Chen, J.-B. Chang e C.-K. Shieh. "Variable-sized map and locality-aware reduce on public-resource grids". Em: *Future Generation Computer Systems* 27.6 (2011), pp. 843–849. DOI: https://doi.org/10.1016/j.future.2010.09.001.

[Sui+16] J. Sui, C. Xu, S.-C. Cheung, W. Xi, Y. Jiang, C. Cao, X. Ma e J. Lu. "Hybrid CPU–GPU constraint checking: Towards efficient context consistency". Em: *Information and Software Technology* 74 (2016), pp. 230–242. DOI: https://doi.org/10.1016/j.infsof.2015.10.003.

[Sve+09] B. Svensson et al. "Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing". Em: *Microprocessors and Microsystems* 33.3 (2009), pp. 161–178. DOI: https://doi.org/10.1016/j.micpro.2008.10.003.

[Tan+14] C.-H. Tang, M.-F. Tsai, S.-H. Chuang, J.-J. Cheng e W.-J. Wang. "Shortest-linkage-based parallel hierarchical clustering on main-belt moving objects of the solar system". Em: *Future Generation Computer Systems* 34 (2014), pp. 26–46. DOI: https://doi.org/10.1016/j.future.2013.12.029.

[Teo+14] G. Teodoro, T. Pan, T. Kurc, J. Kong, L. Cooper, S. Klasky e J. Saltz. "Region templates: Data representation and management for high-throughput image analysis". Em: *Parallel computing* 40.10 (2014), pp. 589–610. DOI: https://doi.org/10.1016/j.parco.2014.09.003.

[Tur+16] W. Turek, J. Stypka, D. Krzywicki, P. Anielski, K. Pietak, A. Byrski e M. Kisiel-Dorohinicki. "Highly scalable Erlang framework for agent-based metaheuristic computing". Em: *Journal of Computational Science* 17 (2016), pp. 234–248. DOI: https://doi.org/10.1016/j.jocs.2016.03.003.

[VD16] M. Vaidya e S. Deshpande. "Critical Study of Performance Parameters on Distributed File Systems using MapReduce". Em: *Procedia Computer Science* 78 (2016), pp. 224–232. DOI: https://doi.org/10.1016/j.procs.2016.02.037.

[VD+00] A. Van Deursen, P. Klint e J. Visser. "Domain-specific languages: An annotated bibliography". Em: *ACM SIGPLAN Notices* 35.6 (2000), pp. 26–36. DOI: https://doi.org/10.1145/352029.352035.

[Van02] M. Vanneschi. "The programming model of ASSIST, an environment for parallel and distributed portable applications". Em: *Parallel computing* 28.12 (2002), pp. 1709–1732. DOI: https://doi.org/10.1016/S0167-8191(02)00188-6.

[Vec+12] C. Vecchiola, R. N. Calheiros, D. Karunamoorthy e R. Buyya. "Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka". Em: *Future Generation Computer Systems* 28.1 (2012), pp. 58–65. DOI: https://doi.org/10.1016/j.future.2011.05.008.

[Vie+17] A. Viebke, S. Memeti, S. Pllana e A. Abraham. "Chaos: a parallelization scheme for training convolutional neural networks on intel xeon phi". Em: *The Journal of Supercomputing* (2017), pp. 1–31. DOI: https://doi.org/10.1007/s11227-017-1994-x.

[Viñ+17] M. Viñas, B. B. Fraguela, D. Andrade e R. Doallo. "High productivity multi-device exploitation with the Heterogeneous Programming Library". Em: *Journal of Parallel and Distributed Computing* 101 (2017), pp. 51–68. DOI: https://doi.org/10.1016/j.jpdc.2016.11.001.

[Vot+95] L. G. Votta, A. Porter e D. Perry. "Experimental Software Engineering: A Report on the State of the Art". Em: *International Conference on Software Engineering*. Vol. 95. 1995, pp. 277–279.

[WM15] M. Wahib e N. Maruyama. "Automated GPU kernel transformations in large-scale production stencil applications". Em: *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*. ACM. 2015, pp. 259–270. DOI: https://doi.org/10.1145/2749246.2749255.

[Wan+15] D. Wang, D. J. Foran, X. Qi e M. Parashar. "HetroCV: Auto-tuning Framework and Runtime for Image Processing and Computer Vision Applications on Heterogeneous Platform". Em: *Parallel Processing Workshops (ICPPW), 2015 44th International Conference on*. IEEE. 2015, pp. 119–128. DOI: https://doi.org/10.1109/ICPPW.2015.21.

[Wan+12] J. Wang, D. Crawl e I. Altintas. "A framework for distributed data-parallel execution in the Kepler scientific workflow system". Em: *Procedia Computer Science* 9 (2012), pp. 1620–1629. DOI: https://doi.org/10.1016/j.procs.2012.04.178.

[Wan+13] P. Wang, J. Wang, Y. Chen e G. Ni. "Rapid processing of remote sensing images based on cloud computing". Em: *Future Generation Computer Systems* 29.8 (2013), pp. 1963–1968. DOI: https://doi.org/10.1016/j.future.2013.05.002.

[Wan+14] Z. Wang, Y. Liu e P. Ma. "A CUDA-enabled parallel implementation of collaborative filtering". Em: *Procedia Computer Science* 30 (2014), pp. 66–74. DOI: https://doi.org/10.1016/j.procs.2014.05.382.

[WJ12] Z. Wei e J. JaJa. "A fast algorithm for constructing inverted files on heterogeneous platforms". Em: *Journal of Parallel and Distributed Computing* 72.5 (2012), pp. 728–738. DOI: https://doi.org/10.1016/j.jpdc.2012.02.005.

[Wil+11] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz e I. Foster. "Swift: A language for distributed parallel scripting". Em: *Parallel Computing* 37.9 (2011), pp. 633–652. DOI: https://doi.org/10.1016/j.parco.2011.05.005.

[Woh14a] C. Wohlin. "Guidelines for snowballing in systematic literature studies and a replication in software engineering". Em: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. ACM. 2014, p. 38. DOI: https://doi.org/10.1145/2601248.2601268.

[Woh14b] C. Wohlin. "Writing for synthesis of evidence in empirical software engineering". Em: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM. 2014, p. 46. DOI: https://doi.org/10.1145/2652524.2652559.

[Yan+12] C. Yan, T. Yue e G Zhao. "GPU Accelerated High Accuracy Surface Modelling". Em: *Procedia Environmental Sciences* 13 (2012), pp. 555–564. DOI: https://doi.org/10.1016/j.proenv.2012.01.046.

[Yan+13] W. Yan, U. Brahmakshatriya, Y. Xue, M. Gilder e B. Wise. "p-PIC: Parallel power iteration clustering for big data". Em: *Journal of Parallel and Distributed computing* 73.3 (2013), pp. 352–359. DOI: https://doi.org/10.1016/j.jpdc.2012.06.009.

[Yil+17] O. Yildiz, S. Ibrahim e G. Antoniu. "Enabling fast failure recovery in shared Hadoop clusters: towards failure-aware scheduling". Em: *Future Generation Computer Systems* 74 (2017), pp. 208–219. DOI: https://doi.org/10.1016/j.future.2016.02.015.

[Zag+15] Q. Zagarese, G. Canfora, E. Zimeo, I. Alshabani, L. Pellegrino, A. Alshabani e F. Baude. "Improving data-intensive EDA performance with annotation-driven laziness". Em: *Science of Computer Programming* 97 (2015), pp. 266–279. DOI: https://doi.org/10.1016/j.scico.2014.03.007.

[Zah+10] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker e I. Stoica. "Spark: Cluster computing with working sets." Em: *HotCloud* 10.10-10 (2010), p. 95. DOI: https://dl.acm.org/citation.cfm?id=1863103.1863113.

[Zha+11] D. Zhang, P. Coddington e A. Wendelborn. "Web services workflow with result data forwarding as resources". Em: *Future Generation Computer Systems* 27.6 (2011), pp. 694–702. DOI: https://doi.org/10.1016/j.future.2010.12.015.

[ZM11] Y. Zhang e F. Mueller. "GStream: A general-purpose data streaming framework on GPU clusters". Em: *Parallel Processing (ICPP), 2011 International Conference on*. IEEE. 2011, pp. 245–254. DOI: https://doi.org/10.1109/ICPP.2011.22.

[Zik+11] P. C. Zikopoulos, C. Eaton, D. Deroos, T. Deutsch e G. Lapis. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 2011. ISBN: 978-0-07-179053-6.

Apêndice A

Questionário

O presente apêndice trata-se de um questionário que formulei, a partir das questões de investigação definidas, que teve como objetivo a recolha da opinião de profissionais da área da CAD. Posteriormente, esses dados foram analisados e comparados com o conhecimento obtido através dos estudos primários incluídos no EMS, de modo a verificar se a informação encontrada na literatura era a esperada.

This survey is being carried out within the scope of the dissertation "Linguagens para a Computação de Alto Desempenho, utilizadas no processamento de *Big Data*: Um Estudo de Mapeamento Sistemático", associated to the student Beatriz Norberto no. 42653, from Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa. For that, it is important to question people within that area.

The questionnaire is of short duration and all your answers are totally confidential.

Thank you for your attention.

1. Are you involved in the SMS?
   - O Yes, I am
   - O No, I am not
2. How long have you been working on High Performance Computing?
   - O Less than 2 years
   - O Between 2 and 5 years
   - O Between 5 and 10 years
   - O More than 10 years
3. In what areas of engineering have you worked? (e.g. Bioinformatics, Telecommunications)

4. Does your High Performance Computing activity consist primarily of developing support tools or of using existing tools?
5. Which programming languages do you use for High Performance Computing?
6. What made you use these languages in relation to the alternatives you know? (this may include language properties and contextual factors, etc)
7. What are the key advantages of these languages?
8. How do you rate the existing tool support for the languages you use for HPC?

   O Very Poor O Poor O Neutral O Good O Excellent

9. In relation to the previous question, what are the existing support tools you know?
10. For the domain where you are, how do you rate the effectiveness of the languages you use?

    O Very Poor O Poor O Neutral O Good O Excellent

11. What are the fundamental language mechanisms that justify your previous answer?
12. What are the performance gains brought by the languages reported?
13. What are the limitations/difficulties of the languages you use?
14. How do you rate your level of technical knowledge for languages used for HPC?

    O Very Poor O Poor O Neutral O Good O Excellent

Apêndice B

Protocolo de Revisão

No presente apêndice, encontram-se listados os componentes de um protocolo de revisão, desenvolvido aquando do Planeamento do EMS.

1. Justificação para a Criação do EMS
2. Questões de Investigação
3. Estratégia de Pesquisa
    a) Termos de Pesquisa
    b) Bibliotecas Digitais
4. Critérios de Seleção dos Estudos
    a) Critérios de Inclusão
    b) Critérios de Exclusão
5. Validação do Processo de Revisão
6. Estratégia de Extração de Dados
7. Estratégia de Sintetização da Informação Recolhida
8. Estratégia de Comunicação (p.e. Revistas Científicas, Conferências)
9. Calendarização do Projeto

Apêndice C

Preenchimento do Formulário de Extração de Informação de um Estudo

No presente apêndice, é apresentado, como exemplo, um formulário de extração de informação que preenchi de um dos estudos incluídos no EMS.

Tabela C.1: Exemplo de preenchimento do Formulário de Extração de Informação

Paper ID	421
Reviewer	Beatriz Norberto
Author	Claudia Misale; Maurizio Drocco; Guy Tremblay; Alberto R. Martinelli; Marco Aldinucci
Title	PiCo: High-performance data analytics pipelines in modern C++
Volume	87
Date of publication	May 2018
Pages	392–403
URL	https://doi.org/10.1016/j.future.2018.05.030

Abstract	"In this paper, we present a new C++ API with a fluent interface called PiCo (Pipeline Composition). PiCo's programming model aims at making easier the programming of data analytics applications while preserving or enhancing their performance. This is attained through three key design choices: (1) unifying batch and stream data access models, (2) decoupling processing from data layout, and (3) exploiting a stream-oriented, scalable, efficient C++11 runtime system. PiCo proposes a programming model based on pipelines and operators that are polymorphic with respect to data types in the sense that it is possible to reuse the same algorithms and pipelines on different data models (e.g., streams, lists, sets, etc.). Preliminary results show that PiCo, when compared to Spark and Flink, can attain better performances in terms of execution times and can hugely improve memory utilization, both for batch and stream processing."
Keywords	Big Data; High Performance Data Analytics; Domain Specific Language; C++; Stream Computing; Fog Computing; Edge Computing
Does the publication include COST CHiPSet's authors?	Yes, Marco Aldinucci
What are the institutions involved?	Cognitive and Cloud, Data-Centric Solutions, IBM T.J. Watson Research Center, Yorktown Heights, NY, USA; Computer Science Department, University of Torino, Torino, Italy; Département d'Informatique, Université du Québec à Montréal, Montréal, QC, Canada
What is name of the conference or journal?	Future Generation Computer Systems
Who is sponsoring the research?	Both private and public funds
What kind of research is being reported?	Case-study; Experiment Report; Comparative Assessment
What kind of language is it?	Embedded DSL in a GPL - A language embedded in a host language

Name of the language?	Pipeline Composition (PiCo)
In case of on an embedded language, what is name of the host language?	C++
What is/are the application domain/s of the language?	Big Data Analytics
What is the purpose of the language?	Formalisation of the solution; Simulation of the solution; Implement the solution; Data interpretation
What are the key advantages of the language?	Performance; Portability; Easiness of configuration; Usability (Effectiveness/Efficiency/Satisfaction)
What are the paradigms underlying the language?	Functional; Object-Oriented
Is there a concrete syntax for the language?	Yes
What are the preferred concrete language syntax (representation) types?	Textual
What is the existing tool support for the language?	Compilers; Tool suite
Which are the execution stack requirements to support the artifacts created with those languages?	**OS** (PiCo application can be compiled to any target platform supporting a modern C++ compiler)
Does the language target specific hardware?	No
Does the language target GPUs or multi-core architectures?	Yes
What are the roles of the users of this language?	End-user

What kind of technical knowledge is required?	**Languages** (C++); **Frameworks** (FastFlow); **Theoretical Background** (Batch and Streaming Applications)
Is the success of the languages evaluated in the paper?	Evaluated
Is there an explicit comparison with competing approaches?	Yes, they have compared PiCo to two state-of-the-art frameworks: Spark and Flink (Execution times)
Is there an explicit comparison of the language proposal with respect to distinct settings/context/configurations?	Yes
What type of comparison if performed?	Quantitative
What is the comparison methodology?	They have compared PiCo to two state-of-the-art frameworks (Spark and Flink) execution times in shared memory for both batch and stream applications
What are the metrics used?	**Time** (Quantitative)
What is the impact on the productivity gains brought by the languages reported?	**Expressiveness** (Qualitative); **Easier to use** (Qualitative)
What is the impact on the products' performance gains brought by the languages reported?	**Memory Efficiency** (Quantitative); **Computation Efficiency** (Quantitative); **Scalability** (Efficiency)

Apêndice D

Sumarização da Informação Recolhida

Tabela D.1: Lista das Linguagens encontradas

Linguagens de Domínio Específico
CineGrid Description Language + Network Description Language
Crucible
e-Science Central WFMS
Higher-order "chemical programming" language
Liszt
Mendeleev
MiniZinc
Linguagens de Propósito Geral
Bobolang
C/C++
Erlang
FastFlow
Goal Language supported by RuGPlanner
Java
OpenCL
Python/R

Scout

Selective Embedded Just-In-Time Specialization

SkIE-CL

Swift

Linguagens de Domínio Específico embebidas em Linguagens de Propósito Geral

Pipeline Composition

Spark Streaming and Spark SQL

Weaver

D.1 Informação relativa à Questão de Investigação 2

Linguagens de Domínio Específico

Tabela D.2: Informação relativa à Questão 2 sobre as linguagens *CineGrid Description Language + Network Description Language* extraída do artigo [Kon+11]

QI 2: Nature of the language	
Ontology languages describing domain-specific services and network entities, for the domain of a non-public digital media data grid, in OWL (i.e., ultimately, XML)	
Purpose of the language	Formalisation of the requirements of the problem; Formalisation of the solution; Data interpretation
Key advantages	Portability, Easiness of configuration, Visualization of user-initiated query results
Paradigms underlying the language	Declarative (Data access service configuration and deployment structure graphs expressed in OWL/XML syntax)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters
Technologies used to create the language tool suite	XML based technology (Jess reasoner for querying OWL ontologies)

Tabela D.3: Informação relativa à Questão 2 sobre a linguagem *Crucible* extraída do artigo [Coe+14]

QI 2: Nature of the language	
Host languages	Java
Application domain	Data analytics
Purpose of the language	Implement the solution
Key advantages	Portability, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Object-Oriented
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters, Compilers, Tool suite
Technologies used to create the language tool suite	IBM Infosphere, Accumulo, HDFS
Execution stack requirements to support the artifacts create with those languages	OS (any), IO architecture (HDFS), Message Passing Middleware (IBM Infosphere)
Execution model that is being used	Virtual Execution Environment (JVM), Distributed Middleware (IBM InfoSphere), Compiled code for CPU

Tabela D.4: Informação relativa à Questão 2 sobre a linguagem *e-Science Central WFMS* extraída do artigo [Cal+16]

QI 2: Nature of the language	
Host languages	Workflow blocks can be written in Java, R, Octave and Javascript
Application domain	Cloud-based data analysis
Purpose of the language	Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
There is a concrete syntax for the language and the preferred representation type is Diagrammatic	
Existing tool support for the language	Tool suite
Technologies used to create the language tool suite	They describe porting of a genomics data processing pipeline from a shell-script implementation on a HPC cluster, to e-Science Central based workflow on Microsoft Azure cloud

Tabela D.5: Informação relativa à Questão 2 sobre a linguagem *Higher-order "chemical programming" language* extraída do artigo [Fer+14]

QI 2: Nature of the language	
Application domain	A rule-based coordination language for asynchronous, self-organizing parallel processing of scientific workflows
Purpose of the language	Formalisation of the solution, Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Declarative (rule-based asynchronous coordination), Hybrid (Atoms of the scripting language are usually written in some sequential HPC language like C)
There is a concrete syntax for the language and the preferred representation type is Textual	

Existing tool support for the language	Interpreters, Compilers
Technologies used to create the language tool suite	HOCL interpreter/JIT plus runtime support extensions for parallel/distributed processing, written in Java
Execution stack requirements to support the artifacts created with those languages	Message Passing Middleware (Java Message Service, ActiveMQ, DAIOS WS (WSDL, SOAP)), Java, HOCL Interpreter
Execution model that is being used	Distributed middleware (Java Message Service, ActiveMQ, DAIOS WS (WSDL, SOAP)), Compiled code for CPU (using a JIT)

Tabela D.6: Informação relativa à Questão 2 sobre a linguagem *Liszt* extraída do artigo [DeV+11]

QI 2: Nature of the language	
A DSL, based on Scala, for solving partial differential equations (PDEs) on unstructured meshes	
Application domain	Constructing mesh-based partial differential equations solvers
Purpose of the language	Implement the solution
Key advantages	Portability, Easiness of configuration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional and Object-Oriented (The Liszt programming environment is based on Scala)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers
The language target specific hardware and GPUs or multi-core architectures	

Tabela D.7: Informação relativa à Questão 2 sobre a linguagem *Mendeleev* extraída do artigo [CJ17]

QI 2: Nature of the language	
Application domain	Data analytics
Purpose of the language	Formalisation of the requirements of the problem, Implement the solution
Key advantages	Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Declarative (Goal-based planning of analytic applications using an abstract model based on a semantically annotated type system)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers, Tool suite
Technologies used to create the language tool suite	Compiler generators (IBM Infosphere Streams; Crucible), Goal-based planning of analytic applications with automatic code generation based on Crucible DSL
Execution stack requirements to support the artifacts created with those languages	IO architecture (HDFS and others), Message Passing Middleware (IBM Infosphere Streams)
Execution model that is being used	Virtual Execution Environment (JVM), Distributed Middleware (IBM InfoSphere), Compiled code for CPU

Tabela D.8: Informação relativa à Questão 2 sobre a linguagem *MiniZinc* extraída do artigo [Cab+15]

QI 2: Nature of the language	
Application domain	Constraint modeling language
Purpose of the language	Formalisation of the requirements of the problem, Formalisation of the solution, Implement the solution
Key advantages	Usability (Effectiveness/Efficiency/Satisfaction), Easier to express constraint problems
Paradigms underlying the language	Hybrid (The constraints are expressed with logic operators)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers, Tool suite, IDE
Technologies used to create the language tool suite	The compiler compiles MiniZinc to FlatZinc, a language that is understood by a wide range of solvers

Linguagens de Propósito Geral

Tabela D.9: Informação relativa à Questão 2 sobre a linguagem *Bobolang* extraída do artigo [Fal+14]

QI 2: Nature of the language	
Specification language for streaming applications	
Application domain	Design of streaming applications
Purpose of the language	Formalisation of the solution, Data interpretation
Key advantages	Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Declarative (it is a specification language dedicated to designing streaming applications)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers
Technologies used to create the language tool suite	Underlying system language (e.g. C++)
Execution model that is being used	Compiled code for CPU (from underlying system language)

Tabela D.10: Informação relativa à Questão 2 sobre as linguagens *C/C++* extraída dos artigos [Bad+15; Bin+13; EK10; Lia+16; Mea+17; Obr+12; Sen+15]

QI 2: Nature of the language	
Application domain	Scientific Computing, Heterogeneous Computing
Purpose of the language	Formalisation of the requirements of the problem, Formalisation of the solution, Simulation of the problem, Simulation of the solution, Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters, Compilers, Validators, Simulators, Tool suite, IDE
Technologies used to create the language tool suite	GenERTiCA source code generator
Execution stack requirements to support the artifacts created with those languages	Multiple OS supported
Execution model that is being used	Virtual Execution Environment (self-managed), Distributed middleware (self-managed), Compiled code for CPU, Compiled code for GPU
The language target GPUs or multi-core architectures	

Tabela D.11: Informação relativa à Questão 2 sobre a linguagem *Erlang* extraída do artigo [Tur+16]

QI 2: Nature of the language	
Application domain	Computational and memory-intensive applications using a high number of cores (64). The use-case is urban traffic planning
Purpose of the language	Implement the solution, Data interpretation
Key advantages	Performance, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters, Compilers, Tool suite, IDE
Execution stack requirements to support the artifacts created with those languages	Message Passing Middleware (Erlang uses a message passing system to communicate between agents), Libraries ("exometer" for global logging and "lcnt" to monitor lock contention)
Execution model that is being used	Virtual Execution Environment (Erlang includes a stack-based VM)
The language target GPUs or multi-core architectures	

Tabela D.12: Informação relativa à Questão 2 sobre a linguagem *FastFlow* extraída dos artigos [Ald+17; Men+17]

QI 2: Nature of the language	
Host languages	C++
Application domain	Streaming applications
Purpose of the language	Implement the solution
Key advantages	Performance, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional, Object-Oriented
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers
The language target GPUs or multi-core architectures	

Tabela D.13: Informação relativa à Questão 2 sobre a linguagem *Goal Language (supported by RuGPlanner)* extraída do artigo [Kal+16]

QI 2: Nature of the language	
A declarative language for expressing extended goals, allows for continual plan revision to deal with sensing outputs, failures, long response times or timeouts, as well as the activities of external agents; Many elements of the language are inspired by XSRL (XML Service Request Language)	
Purpose of the language	Formalisation of the requirements of the problem, Formalisation of the solution, Implement the solution, Data interpretation
Key advantages	Performance, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Declarative (Provides the user with expressive constructs for stating complex goals, beyond the mere statement of properties that should hold in the final state), Functional (comprises a number of atomic service operations that can serve a variety of objectives with minimal request-specific configuration), Logic (it is based on translating the domain and the goal into a Constraint Satisfaction Problem)
There is a concrete syntax for the language and the preferred representation type is Textual	
Technologies used to create the language tool suite	An extended language detached from the particularities and interdependencies of the available services
Execution model that is being used	Compiled code for CPU

Tabela D.14: Informação relativa à Questão 2 sobre a linguagem *Java* extraída dos artigos [Bad+15; Car+13; Mat+10; Mat+11; Mea+17]

QI 2: Nature of the language	
Application domain	Grid applications to Ray tracing and Sequencing; Machine Learning; Specify policies to transform divide and conquer sequential programs into parallel executions
Purpose of the language	Formalisation of the requirements of the problem, Formalisation of the solution, Simulation of the solution, Implement the solution, Data interpretation
Key advantages	Performance, Portability, Easiness of configuration, Orchestration e Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Object-Oriented, Hybrid (Language to schedule constraint solving)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters, Compilers
Technologies used to create the language tool suite	XML based technology (A XML like syntax to describe classes and methods to be scheduled)
Execution stack requirements to support the artifacts created with those languages	VM Supervisor (JVM on grid), OS (any), IO architecture (Grid), Libraries (Spark, 77 Weka 3.6.0, Hadoop 0.20)
Execution model that is being used	Virtual Execution Environment (Java Virtual Machine), Distributed middleware (Hadoop, Spark), HPC Libraries (Spark), Bytecode for virtual machine (JVM on Grid)

Tabela D.15: Informação relativa à Questão 2 sobre a linguagem *OpenCL* extraída dos artigos [Bin+13; Kim+15]

QI 2: Nature of the language	
Application domain	CFD (any application that benefits from GPU), Big Data processing
Purpose of the language	Formalisation of the requirements of the problem, Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Object-Oriented
There is a concrete syntax for the language and the preferred representation type can be both Textual and Diagrammatic	
Existing tool support for the language	Compilers, Tool suite
Technologies used to create the language tool suite	GenERTiCA source code generator
Execution stack requirements to support the artifacts created with those languages	Multiple OS supported
Execution model that is being used	Distributed middleware, HPC Libraries, Bytecode for virtual machine, Compiled code for CPU, Compiled code for GPU
The language target specific hardware and GPUs or multi-core architectures	

Tabela D.16: Informação relativa à Questão 2 sobre as linguagens *Python/R* extraída dos artigos [Bad+15; Hin+06; Luc+15]

QI 2: Nature of the language	
Application domain	High-level parallel programming language for scientific computing, distributed applications
Purpose of the language	Formalisation of the requirements of the problem, Formalisation of the solution, Simulation of the problem, Simulation of the solution, Implement the solution, Data interpretation
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Supports multiple programming paradigms
There is a concrete syntax for the language and the preferred representation type is both Textual and Diagrammatic	
Existing tool support for the language	Interpreters, Compilers, Validators, Simulators, Tool suite, IDE
Execution stack requirements to support the artifacts created with those languages	OS (Any), Message Passing Middleware (BSP model), Libraries
Execution model that is being used	Virtual Execution Model (self-managed), Distributed Middleware (self-managed), Compiled code for CPU

Tabela D.17: Informação relativa à Questão 2 sobre a linguagem *Scout* extraída do artigo [McC+07]

QI 2: Nature of the language	
Purpose of the language	Formalisation of the solution, Implement the solution, Data interpretation
Key advantages	Portability, Easiness of configuration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Object-Oriented (the base language from which Scout extends is C*, which is object-oriented)

There is a concrete syntax for the language and the preferred representation type is Textual

Existing tool support for the language	Compilers

The language target specific hardware and GPUs or multi-core architectures

Tabela D.18: Informação relativa à Questão 2 sobre a linguagem *Selective Embedded Just-In-Time Specialization* extraída do artigo [Lug+15]

QI 2: Nature of the language	
Host languages	Knowledge Discovery Toolbox (KDT)
Application domain	Semantic Graphs
Purpose of the language	Graph Processing (Implement the solution)
Key advantages	Performance, Easiness of configuration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional, Object-Oriented

There is a concrete syntax for the language and the preferred representation type is Textual

Existing tool support for the language	Interpreters, Compilers, Tool suite
Technologies used to create the language tool suite	DSL frameworks (KDT), compBLAS library
Execution stack requirements to support the artifacts created with those languages	OS (any), Message Passing Middleware (MPI), Libraries (compBLAS)
Execution model that is being used	HPC Libraries (compBLAS), Compiled code for CPU

Tabela D.19: Informação relativa à Questão 2 sobre a linguagem *SkIE-CL* extraída do artigo [CV02]

QI 2: Nature of the language	
SkIE-CL, the programming language of the SkIE (SkIE stands for skeleton integrated environment) environment	
Host languages	C/C++, Java
Application domain	Data mining
Purpose of the language	Implement the solution
Key advantages	Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction), Enables high-level parallel programming using skeletons
Paradigms underlying the language	Skeletons are used as basic constructs of coordination language (SkIE-CL)
There is a concrete syntax for the language and the preferred representation type is both Textual and Diagrammatic	
Existing tool support for the language	Compilers, Tool suite and IDE
Execution stack requirements to support the artifacts created with those languages	OS (Multiple: Linux, Sun Solaris...), Message Passing Middleware (MPI)
Execution model that is being used	Compiled code for CPU

Tabela D.20: Informação relativa à Questão 2 sobre a linguagem *Swift* extraída dos artigos [Mah+16; Wil+11]

QI 2: Nature of the language	
Application domain	Parallel workflow/Distributed parallel scripting
Purpose of the language	Implement the solution
Key advantages	Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional (application components modelled as side-effect free functions)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Interpreters, Tool suite
Execution stack requirements to support the artifacts created with those languages	OS (Linux), IO architecture (POSIX), Message Passing Middleware (Globus)
Execution model that is being used	Virtual Execution Environment (Cloud), Distributed Middleware (Globus Grid middleware)

Linguagens de Domínio Específico embebidas em Linguagens de Propósito Geral

Tabela D.21: Informação relativa à Questão 2 sobre a linguagem *Pipeline Composition (PiCo)* extraída do artigo [Mis+18]

QI 2: Nature of the language	
Host language	C++
Application domain	Big Data Analytics
Purpose of the language	Formalisation of the solution, Simulation of the solution, Implement the solution, Data interpretation
Key advantages	Performance, Portability, Easiness of configuration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional, Object-Oriented
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers, Tool suite
Execution stack requirements to support the artifacts created with those languages	OS (PiCo application can be compiled to any target platform supporting a modern C++ compiler)
The language target GPUs or multi-core architectures	

Tabela D.22: Informação relativa à Questão 2 sobre as linguagens *Spark Streaming* e *Spark SQL* extraída do artigo [Liu+15]

QI 2: Nature of the language	
Host language	Spark applications can be written in Java, Scala, Python, R
Application domain	Streaming analytics
Purpose of the language	Simulation of the problem, Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Orchestration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional (Scala), Object-Oriented (Scala)

There is a concrete syntax for the language and the preferred representation type is Textual

Existing tool support for the language	Compilers
Execution stack requirements to support the artifacts created with those languages	OS (Linux, MS Windows, macOS), IO architecture (Spark Core), Libraries (MLlib Machine Learning Library)
Execution model that is being used	Distributed Middleware (Hadoop Distributed File System (HDFS), OpenStack Swift,..)

The language target GPUs or multi-core architectures

Tabela D.23: Informação relativa à Questão 2 sobre a linguagem *Weaver* extraída do artigo [Bui+11]

QI 2: Nature of the language	
A DSL built on top of Python which allows researchers to construct scalable scientific data-processing workflows	
Host language	Python
Application domain	Scientific workflows
Purpose of the language	Formalisation of the solution, Implement the solution
Key advantages	Performance, Portability, Easiness of configuration, Usability (Effectiveness/Efficiency/Satisfaction)
Paradigms underlying the language	Functional and Object-Oriented (built on top of Python)
There is a concrete syntax for the language and the preferred representation type is Textual	
Existing tool support for the language	Compilers, Tool suite

D.2 Informação relativa à Questão de Investigação 3

Linguagens de Domínio Específico

Tabela D.24: Informação relativa à Questão 3 sobre as linguagens *CineGrid Description Language + Network Description Language* extraída do artigo [Kon+11]

QI 3: Typical user profiles for the language	
Role of the users of this language	Developers
Technical knowledge required	Tools (OWL/XML editor), Languages (SQWRL query language for OWL ontologies), Hardware/Systems (Data grids), Theoretical Background (XML database querying and reasoning)

Tabela D.25: Informação relativa à Questão 3 sobre a linguagem *Crucible* extraída do artigo [Coe+14]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Tools (XText), Languages (Java), Frameworks (IBM Infosphere), Hardware (CPU), Systems (Clusters), Theoretical Background (Communicating Sequential Processes)

Tabela D.26: Informação relativa à Questão 3 sobre a linguagem *e-Science Central WFMS* extraída do artigo [Cal+16]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (workflow), Systems (Amazon AWS, Microsoft Azure)

Tabela D.27: Informação relativa à Questão 3 sobre a linguagem *Higher-order "chemical programming" language* extraída do artigo [Fer+14]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Java, "chemical programming" in HOCL), Theoretical Background (Rule-based programming, "chemical programming" for WS/workflow coordination)

Tabela D.28: Informação relativa à Questão 3 sobre a linguagem *Liszt* extraída do artigo [DeV+11]

QI 3: Typical user profiles for the language	
Technical knowledge required	Languages (Scala)

Tabela D.29: Informação relativa à Questão 3 sobre a linguagem *Mendeleev* extraída do artigo [CJ17]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Tools (Mendeleev DSL), Languages (RDF, IBM InfoSphere, Accumulo), Frameworks (Crucible, IBM Infosphere), Hardware (CPU), Systems (Clusters), Theoretical Background (RDF graphs)

Tabela D.30: Informação relativa à Questão 3 sobre a linguagem *MiniZinc* extraída do artigo [Cab+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Theoretical Background (Constraint modelling)

Linguagens de Propósito Geral

Tabela D.31: Informação relativa à Questão 3 sobre a linguagem *Bobolang* extraída do artigo [Fal+14]

QI 3: Typical user profiles for the language	
Role of the users of this language	Developers
Technical knowledge required	Theoretical Background (Domain of streaming applications)

Tabela D.32: Informação relativa à Questão 3 sobre as linguagens *C/C++* extraída dos artigos [Bad+15; Bin+13; EK10; Lia+16; Mea+17; Obr+12; Sen+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users and Developers
Technical knowledge required	Languages (C/C++), Hardware (parallel and distributed systems; Grids; Clouds)

Tabela D.33: Informação relativa à Questão 3 sobre a linguagem *Erlang* extraída do artigo [Tur+16]

QI 3: Typical user profiles for the language	
Technical knowledge required	Languages (Erlang), Theoretical Background (Agent-oriented frameworks and Evolutionary systems)

Tabela D.34: Informação relativa à Questão 3 sobre a linguagem *FastFlow* extraída do artigo [Ald+17; Men+17]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (C++), Hardware (CPU), Theoretical Background (Streaming Applications)

Tabela D.35: Informação relativa à Questão 3 sobre a linguagem *Goal Language (supported by RuGPlanner)* extraída do artigo [Kal+16]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Goal language)

Tabela D.36: Informação relativa à Questão 3 sobre a linguagem *Java* extraída dos artigos [Bad+15; Car+13; Mat+10; Mat+11; Mea+17]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Java)

Tabela D.37: Informação relativa à Questão 3 sobre a linguagem *OpenCL* extraída dos artigos [Bin+13; Kim+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Tools (detailed knowledge required for using OpenCL for GPUs), Languages (OpenCL), Hardware (Clusters with GPUs)

Tabela D.38: Informação relativa à Questão 3 sobre as linguagens *Python/R* extraída dos artigos [Bad+15; Hin+06; Luc+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users and Developers
Technical knowledge required	Languages (Python/R), Hardware (parallel and distributed systems; Grids; Clouds)

Tabela D.39: Informação relativa à Questão 3 sobre a linguagem *Selective Embedded Just-In-Time Specialization* extraída do artigo [Lug+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Python, C++), Libraries (KDT), Hardware (CPU), Systems (Clusters), Theoretical Background (Graph Algorithms)

Tabela D.40: Informação relativa à Questão 3 sobre a linguagem *SkIE-CL* extraída do artigo [CV02]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Tools (Visual SkIE), Languages (SkIE-CL), Theoretical Background (Skeletons)

Tabela D.41: Informação relativa à Questão 3 sobre a linguagem *Swift* extraída dos artigos [Mah+16; Wil+11]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Swift)

Linguagens de Domínio Específico embebidas em Linguagens de Propósito Geral

Tabela D.42: Informação relativa à Questão 3 sobre a linguagem *Pipeline Composition (PiCo)* extraída do artigo [Mis+18]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (C++), Frameworks (FastFlow), Theoretical Background (Batch and Streaming Applications)

Tabela D.43: Informação relativa à Questão 3 sobre as linguagens *Spark Streaming* e *Spark SQL* extraída do artigo [Liu+15]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Frameworks (Spark)

Tabela D.44: Informação relativa à Questão 3 sobre a linguagem *Weaver* extraída do artigo [Bui+11]

QI 3: Typical user profiles for the language	
Role of the users of this language	End-users
Technical knowledge required	Languages (Python)

D.3 Informação relativa à Questão de Investigação 4

Linguagens de Domínio Específico

Tabela D.45: Informação relativa à Questão 4 sobre as linguagens *CineGrid Description Language + Network Description Language* extraída do artigo [Kon+11]

QI 4: Effectiveness of the language
Success of the language not evaluated

Tabela D.46: Informação relativa à Questão 4 sobre a linguagem *Crucible* extraída do artigo [Coe+14]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches, Quantitative comparison performed	
Productivity gains brought by the languages	Expressiveness, Easier to use - Qualitative
Products' performance gains brought	Evolvability/Maintainability - Qualitative

Tabela D.47: Informação relativa à Questão 4 sobre a linguagem *e-Science Central WFMS* extraída do artigo [Cal+16]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Compared shell-script implementation on a HPC cluster with workflow on Microsoft Azure cloud	
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Easier to use - Qualitative and e-Science Central enables users to design workflows for data analysis
Products' performance gains brought	Computation efficiency, Scalability - Quantitative; Evolvability/Maintainability - Qualitative

Tabela D.48: Informação relativa à Questão 4 sobre a linguagem *Higher-order "chemical programming" language* extraída do artigo [Fer+14]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Experimental comparison with two traditional-style workflow systems based on 3 HPC test problems	
Metrics	Time
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Expressiveness, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency - Quantitative; Evolvability/-Maintainability, Scalability - Qualitative

Tabela D.49: Informação relativa à Questão 4 sobre a linguagem *Liszt* extraída do artigo [DeV+11]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
The authors ported four example applications to Liszt and ran these applications on three platforms: a GPU, a SMP and a cluster. They evaluate the MPI-based runtime on both the cluster and the SMP since it can run on either platform	
Metrics	Lines of Code, Time
Products' performance gains brought	Computation efficiency, Scalability - Quantitative; Memory Efficiency - Qualitative

Tabela D.50: Informação relativa à Questão 4 sobre a linguagem *Mendeleev* extraída do artigo [CJ17]

QI 4: Effectiveness of the language
Success of the language evaluated

Tabela D.51: Informação relativa à Questão 4 sobre a linguagem *MiniZinc* extraída do artigo [Cab+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Both Quantitative and Qualitative comparison performed	
The paper compares base version of MiniZinc with one integrating the extensions	
Metrics	Lines of Code, Time
Productivity gains brought by the languages	Expressiveness - Qualitative, Easier to use - Quantitative
Products' performance gains brought	Memory efficiency, Computation efficiency - Quantitative

Linguagens de Propósito Geral

Tabela D.52: Informação relativa à Questão 4 sobre a linguagem *Bobolang* extraída do artigo [Fal+14]

QI 4: Effectiveness of the language
Success of the language not evaluated

Tabela D.53: Informação relativa à Questão 4 sobre as linguagens *C/C++* extraída dos artigos [Bad+15; Bin+13; EK10; Lia+16; Mea+17; Obr+12; Sen+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Algorithms for task scheduling are evaluated	
Metrics	Time
Productivity gains brought by the languages	Learnability - Quantitative and Lower cognitive overload, Easier to remember, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency, Scalability - Quantitative and Evolvability/Maintainability, Scalability - Qualitative

Tabela D.54: Informação relativa à Questão 4 sobre a linguagem *Erlang* extraída do artigo [Tur+16]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison of the language proposal with respect to distinct settings/context/configurations, Quantitative comparison performed	
Scalability of the different techniques when increasing the number of cores	
Metrics	Number of agent reproductions

Tabela D.55: Informação relativa à Questão 4 sobre a linguagem *FastFlow* extraída do artigo [Ald+17; Men+17]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
The applicability of FastFlow has been illustrated by a number of studies in different application domains including image processing, file compression and stochastic simulation	
Metrics	Time
Products' performance gains brought	Memory Efficiency, Computation Efficiency - Quantitative

Tabela D.56: Informação relativa à Questão 4 sobre a linguagem *Goal Language (supported by RuGPlanner)* extraída do artigo [Kal+16]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed, Explicit comparison of the language proposal with respect to distinct settings/context/configurations	
Two test cases. They performed a number of tests regarding the scalability of the system with respect to a number of factors	
Metrics	Lines of code, Satisfaction, Time
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Expressiveness, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency, Scalability - Quantitative

Tabela D.57: Informação relativa à Questão 4 sobre a linguagem *Java* extraída dos artigos [Bad+15; Car+13; Mat+10; Mat+11; Mea+17]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Metrics	Lines of Code, Time
Productivity gains brought by the languages	Easier to use, Compact representation
Products' performance gains brought	Computation efficiency, Scalability - Quantitative

Tabela D.58: Informação relativa à Questão 4 sobre a linguagem *OpenCL* extraída dos artigos [Bin+13; Kim+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed Algorithms for task scheduling are evaluated	
Metrics	Time
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency - Quantitative and Evolvability/-Maintainability - Qualitative

Tabela D.59: Informação relativa à Questão 4 sobre as linguagens *Python/R* extraída dos artigos [Bad+15; Hin+06; Luc+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches, Quantitative comparison performed	
Metrics	Time
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency, Scalability - Quantitative and Scalability - Qualitative

Tabela D.60: Informação relativa à Questão 4 sobre a linguagem *Scout* extraída do artigo [McC+07]

QI 4: Effectiveness of the language	
Success of the language evaluated	
Productivity gains brought by the languages	Lower cognitive overload, Easier to use - Qualitative

Tabela D.61: Informação relativa à Questão 4 sobre a linguagem *Selective Embedded Just-In-Time Specialization* extraída do artigo [Lug+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches and language proposal with respect to distinct settings/context/configurations, Quantitative comparison performed	
Performance and coding complexity evaluation against direct usage of Python interface of KDT and direct usage of KDT backend (i.e. compBLAS) on standard graph algorithms and synthetic datasets (in-core)	
Metrics	Lines of Code, Satisfaction, Time
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Expressiveness, Easier to use - Qualitative
Products' performance gains brought	Memory Efficiency, Computation Efficiency, Scalability - Quantitative and Evolvability/Maintainability - Qualitative

Tabela D.62: Informação relativa à Questão 4 sobre a linguagem *SkIE-CL* extraída do artigo [CV02]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches and language proposal with respect to distinct settings/context/configurations, Quantitative comparison performed	
The language is compared with MPI with respect to number of lines of code and development time	
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to use - Qualitative
Products' performance gains brought	Evolvability/Maintainability - Qualitative; Scalability - Quantitative

Tabela D.63: Informação relativa à Questão 4 sobre a linguagem *Swift* extraída dos artigos [Mah+16; Wil+11]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Metrics	Time, Utilization
Productivity gains brought by the languages	Learnability, Lower cognitive overload, Easier to remember, Expressiveness, Easier to use - Quantitative and Qualitative
Products' performance gains brought	Computation efficiency, Evolvability/Maintainability, Scalability, Resource utilization - Quantitative and Qualitative

Linguagens de Domínio Específico embebidas em Linguagens de Propósito Geral

Tabela D.64: Informação relativa à Questão 4 sobre a linguagem *Pipeline Composition (PiCo)* extraída do artigo [Mis+18]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches (They have compared PiCo to two state-of-the-art frameworks: Spark and Flink) and language proposal with respect to distinct settings/context/configurations, Quantitative comparison performed	
They have compared PiCo to two state-of-the-art frameworks (Spark and Flink) execution times in shared memory for both batch and stream applications	
Metrics	Time
Productivity gains brought by the languages	Expressiveness, Easier to use - Qualitative
Products' performance gains brought	Memory Efficiency, Computation efficiency, Scalability - Quantitative

Tabela D.65: Informação relativa à Questão 4 sobre as linguagens *Spark Streaming* e *Spark SQL* extraída do artigo [Liu+15]

QI 4: Effectiveness of the language	
Success of the language evaluated, Quantitative comparison performed	
Presented experimental results for three datasets	
Metrics	Time
Products' performance gains brought	Computation efficiency, Scalability - Quantitative

Tabela D.66: Informação relativa à Questão 4 sobre a linguagem *Weaver* extraída do artigo [Bui+11]

QI 4: Effectiveness of the language	
Success of the language evaluated, Explicit comparison with competing approaches and language proposal with respect to distinct settings/context/configurations	
They provided four applications constructed using Weaver and evaluated its effectiveness in the context of scripting scientific workflows for distributed systems	
Metrics	Lines of Code, Time
Productivity gains brought by the languages	Learnability, Easier to use - Qualitative
Products' performance gains brought	Computation efficiency, Scalability - Quantitative and Evolvability/Maintainability - Qualitative

Apêndice E

Lista de Artigos Incluídos

No presente apêndice, é apresentada a lista de artigos que foram incluídos no EMS e analisados, respondendo às questões de investigação identificadas.

Tabela E.1: Lista de artigos incluídos na revisão

ID da Publicação	Referência da Publicação
P003	[Son+12]
P004	[Siv+17]
P006	[Su+11]
P007	[Tan+14]
P008	[Sim+15]
P010	[Sui+16]
P018	[Lee+09]
P023	[BC17]
P030	[Rat16]
P031	[Pin+15]
P046	[Jin+17]
P049	[Dee+09]
P051	[Ded+14]
P052	[CV02]
P055	[Mat+11]
P056	[Mat+10]

P060	[Cra15]
P062	[Ma+14]
P064	[Ma+15]
P065	[Mah+16]
P066	[Han+11]
P070	[HN15a]
P073	[CP16]
P076	[KS15]
P077	[Gia15]
P078	[Giu+11]
P079	[Kon+11]
P080	[Fer+14]
P082	[Kra+17]
P084	[Kol+13]
P085	[Kra+15]
P086	[Gil+13]
P090	[Fer+16]
P093	[Hin+06]
P095	[Hsu14]
P096	[Dee+15]
P097	[Hsu+15]
P101	[Núñ+12]
P102	[Fer+15]
P103	[EP16]
P104	[Mou+14]
P105	[Dia+15]
P106	[Pic+15]
P107	[PN14]
P109	[Lia+07]
P110	[Buy+09]
P111	[BL15]
P114	[LG+16]

P115	[Pap+17]
P117	[Bur+16]
P118	[Bin+13]
P119	[Lia+16]
P120	[Liu+15]
P134	[Obr+12]
P135	[Oos+15]
P138	[Cab+15]
P139	[Cal+16]
P141	[Luc+15]
P142	[Pan+17]
P143	[Obr+11]
P145	[Car+13]
P146	[Cav+14]
P147	[Err+15]
P149	[Guh+09]
P150	[Erä+14]
P151	[Mea+17]
P153	[Gun+13]
P154	[Gub+13]
P155	[Mic+17]
P156	[McC+07]
P157	[May+11]
P158	[Mor+07]
P159	[Gon+15]
P160	[Gra+15]
P161	[EO+16]
P162	[Kal+16]
P163	[Sve+09]
P164	[Gar+08]
P165	[Fol+10]
P166	[Sec+10]

P167	[Kaj+14]
P168	[For+14]
P169	[Alb+15]
P171	[SR14]
P172	[Aji+16]
P176	[Sch+15]
P177	[GH07]
P178	[KP13]
P179	[Sch+11]
P180	[Kal13]
P181	[Sat15]
P182	[Ser+16]
P184	[He+16]
P186	[Alh+11]
P188	[HN15b]
P190	[AD08]
P192	[Hua+17]
P194	[Alm+14]
P195	[Hün+15]
P196	[Han+15]
P199	[Kim+15]
P200	[Ros+16]
P203	[Anj+15]
P204	[Kat+16]
P209	[Bar+10]
P211	[Bad+15]
P230	[Sec+17]
P234	[Zag+15]
P235	[Zha+11]
P236	[Vec+12]
P237	[Wan+13]
P238	[Yan+12]

P243	[Yil+17]
P246	[Van02]
P247	[Wil+11]
P248	[Teo+14]
P250	[WJ12]
P251	[Yan+13]
P257	[Wan+12]
P258	[Wan+14]
P261	[VD16]
P262	[Tur+16]
P274	[Kaj+14]
P359	[Sen+15]
P360	[Sal+16]
P361	[Coe+14]
P364	[CJ17]
P365	[Lug+15]
P367	[Viñ+17]
P370	[ZM11]
P371	[Wan+15]
P372	[Fal+14]
P374	[WM15]
P375	[Hij+15]
P376	[Beh+16]
P377	[Edm+13]
P378	[Cha+13]
P379	[Che+15]
P380	[Bui+11]
P383	[DeV+11]
P387	[SB16]
P388	[Li+15]
P412	[Mis+17]
P413	[Men+17]

P414	[EK10]
P415	[Sjö+16]
P416	[Ald+17]
P417	[DS+17]
P418	[Vie+17]
P419	[Ben+11]
P420	[Mem+17]
P421	[Mis+18]

ANEXO I

Implementação de um Grafo em *Java*

O presente anexo trata-se da implementação de um grafo, podendo este ser direcionado ou não direcionado, dependendo do valor da variável booleana *directed*. Este código foi retirado de [Jav].

```java
import java.util.ArrayList;
import java.util.HashMap;

public class Graph<V> {

	private HashMap<V, ArrayList<Edge<V>>> adjacencyList;

	/**
	 * This list holds all the vertices so that we can iterate over them in the
	 * toString function
	 */
	private ArrayList<V> vertexList;

	private boolean directed;

	public Graph(boolean isDirected) {
		directed = isDirected;
		adjacencyList = new HashMap<V, ArrayList<Edge<V>>>();
		vertexList = new ArrayList<V>();
	}

	public void add(V vertex, ArrayList<Edge<V>> connectedVertices) {
		// Add the new vertex to the adjacencyList with it's list of connected
		// nodes
		adjacencyList.put(vertex, connectedVertices);
		vertexList.add(vertex);
		// If this is an undirected graph, every edge needs to represented
		// twice, once in the added vertex's list and once in the list of each
		// of the vertex's connected to the added vertex

			for (Edge<V> vertexConnectedToAddedVertex : connectedVertices) {
				ArrayList<Edge<V>> correspondingConnectedList = adjacencyList
							.get(vertexConnectedToAddedVertex.getVertex());
				// The added vertex's connections might not be represented in
				// the Graph yet, so we implicitly add them
				if (correspondingConnectedList == null) {
					adjacencyList.put(vertexConnectedToAddedVertex.getVertex(),
								new ArrayList<Edge<V>>());
					vertexList.add(vertexConnectedToAddedVertex.getVertex());
					correspondingConnectedList = adjacencyList
								.get(vertexConnectedToAddedVertex.getVertex());
				}

				if (!directed) {
					// The weight from one vertex back to another in an undirected
					// graph is equal
					int weight = vertexConnectedToAddedVertex.getWeight();
					correspondingConnectedList.add(new Edge<V>(vertex, weight));
```

Figura I.1: Implementação de um grafo, na linguagem *Java* [Jav]

```
                        }
                }

}

public boolean addArc(V source, V end, int weight) {
        if (!directed) {
                return false;
        }

        if (!adjacencyList.containsKey(source)) {
                ArrayList<Edge<V>> tempList = new ArrayList<Edge<V>>();
                tempList.add(new Edge<V>(end, weight));
                add(source, tempList);
                return true;
        }

        if (!adjacencyList.containsKey(end)) {
                ArrayList<Edge<V>> tempList = new ArrayList<Edge<V>>();
                add(end, tempList);
        }


        adjacencyList.get(source).add(new Edge<V>(end, weight));
        return true;
}

public boolean addEdge(V vertexOne, V vertexTwo, int weight) {
        if (directed) {
                return false;
        }

        if (!adjacencyList.containsKey(vertexOne)) {
                ArrayList<Edge<V>> tempList = new ArrayList<Edge<V>>();
                tempList.add(new Edge<V>(vertexTwo, weight));
                add(vertexOne, tempList);
                return true;
        }

        if (!adjacencyList.containsKey(vertexTwo)) {
                ArrayList<Edge<V>> tempList = new ArrayList<Edge<V>>();
                tempList.add(new Edge<V>(vertexOne, weight));
                add(vertexTwo, tempList);
                return true;
        }

        adjacencyList.get(vertexOne).add(new Edge<V>(vertexTwo, weight));
        adjacencyList.get(vertexTwo).add(new Edge<V>(vertexOne, weight));
```

Figura I.2: Implementação de um grafo, na linguagem *Java* [Jav] (Continuação)

```
        return true;
}

/**
 * This method returns a list of all adjacent vertices of the give vertex without weight
 *
 * @param vertex the source vertex
 * @return an array list containing the vertices
 */
public ArrayList<V> getAdjacentVertices(V vertex){
        ArrayList<V> returnList = new ArrayList<V>();
        for (Edge<V> edge : adjacencyList.get(vertex)) {
                returnList.add(edge.getVertex());
        }
        return returnList;
}

public double getDistanceBetween(V source, V end){
         for (Edge<V> edge : adjacencyList.get(source)) {
                if (edge.getVertex() == end){
                        return edge.getWeight();
                }
        }
        return Double.POSITIVE_INFINITY;
}

public ArrayList<V> getVertexList() {
        return vertexList;
}

public String toString() {
        String s = "";
        for (V vertex : vertexList) {
                s += vertex.toString();
                s += " : ";
                s += adjacencyList.get(vertex);
                s += "\n";
        }
        return s;
}
```

Figura I.3: Implementação de um grafo, na linguagem *Java* [Jav] (Continuação)